Modelling Mechanisms of Change in Crop Populations

Paul L. R. Partner

Thesis submitted in fulfilment of the degree of Doctor of Philosophy
University of Edinburgh, September 1994

"*Camelot!*"

"*Camelot!*"

"*Camelot!*"

"*It's only a model.*"

"*Shhhhh*!"

From the film, *Monty Python and The Holy Grail* (1974)

DECLARATION

This work is my own. Where others have contributed, this has been acknowledged within the text.

Paul Partner

Edinburgh, 26th September 1994.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

This research was funded by a U.K. Science and Engineering Research Council Co-operative Award in Science and Engineering (CASE), reference number 89101009, in conjunction with Levington Agriculture Ltd., The Norsk Hydro Agricultural Group. Essential research equipment was generously provided by Nicholas M. Langford, friend and patron to the sciences, and latterly by Sandy Gibson and the Computer Services Unit, SAC Edinburgh, who were foolhardy enough to offer me employment.

I am grateful to Drs. M.L Smith and W. Spoor of The Scottish Agricultural College, and to Dr. Ian Richards of Levington Agriculture Ltd., for their guidance and supervision. I am particularly indebted to Mike Smith for maintaining that I was not the worst Ph.D. candidate he'd ever seen. I would also like to thank Robert Redpath and his technical staff, and fellow sufferer Richard Hopkins, for invaluable advice and assistance with the messy stuff, from making mud-pies to the active discouragement of experiment-eating creepy-crawlies.

My thanks go to Neil McRoberts for his guidance and support, and for leading me slowly and patiently through numerous academic minefields; to Drs. Jeff Wagenet, John Hutson and colleagues, Cornell University, for their encouragement and for the wonderful opportunity to conduct part of my research in America; to Bruce Reisch of Cornell University and Hans Haldrup of Pajbjergfonden for their contributions to my education and for their impeccable hospitality; to Frank Wright and Mike Clarkson for their many important technical and social contributions throughout the project; to Professor N.W. Simmonds for his perceptive and helpful comments, and to Professor D.S. Falconer for taking an interest in my selection programme. My worth as a scientist has been greatly enhanced through the opportunity to exchange ideas with Gareth Hughes, football fan.

I would not have been able to last the course were it not for the support and encouragement of my family and friends, whose many and varied contributions included feeding me, cheering me up when I felt down, providing somewhere for me to live, pouring beer down my neck whenever I showed dangerous signs of maturity or other undesirable behaviour, and generally being abnormally tolerant of my temporary insanity and antisocial behaviour. To my parents and my friends, thank you. Finally, my special thanks go Yvonne van Gessel for her constant encouragement and misguided belief in my ability. My life was enriched by your companionship, and I am a better person for knowing how to groom a llama.

In the event of success I would like to dedicate this work to the memory of my grandmother, Mrs. Ruby Jenkins, who always told me I would amount to something. I hope I have not disappointed anyone.

ABSTRACT

Computer-based simulation models of changes occurring within crop populations when subjected to agents of phenotypic change, have been developed for use on commonly available personal computer equipment. As an underlying developmental principle, the models have been designed as general-case, mechanistic, stochastic models, in contrast to the predominantly empirically-derived, system-specific, deterministic (predictive) models currently available. A modelling methodology has evolved, to develop portable simulation models, written in high-level, general purpose code, allowing for use, modification and continued development by biologists with little requirement for computer programming expertise.

The initial subject of these modelling activities was the simulation of the effects of selection and other agents of genetic change in crop populations, resulting in the computer model, PSELECT. Output from PSELECT, specifically phenotypic and genotypic response to phenotypic truncation selection, conformed to expectation, as defined by results from established analogue modelling work. Validation of the model by comparison of output with the results from an experimental-scale plant breeding exercise was less conclusive, and, owing to the fact that the genetic basis of the phenotypic characters used in the selection programme was insufficiently defined, the validation exercise provided only broad qualitative agreement with the model output. By virtue of the predominantly subjective nature of plant breeding programmes, the development of PSELECT resulted in a model of theoretical interest, but with little current practical application.

Modelling techniques from the development of the PSELECT model were applied to the simulation of plant disease epidemics, where the modelled system is well characterised, and simulation modelling is an area of active research. The model SATSUMA, simulating the spatial and temporal development of diseases within crop populations, was developed. The model generates output which conforms to current epidemiological theory, and is compatible with contemporary methods of temporal and spatial analysis of crop disease epidemics. Temporal disease progress in the simulations was accurately described by variations of a generalised logistic model. Analysis of the spatial pattern of simulated epidemics by frequency distribution fitting or distance class methods was found to give good qualitative agreement with observed biological systems.

The mechanistic nature of SATSUMA and its deliberate design as a general case model make it especially suitable for the investigation of component processes in a generalised plant disease epidemic, and valuable as an educational tool. Subject to validation against observational data, such models can be utilised as predictive tools by the incorporation of information (concerning crop species, pathogen etc.) specifically relevant to the modelled system. In addition to its educational use, SATSUMA has been used as research tool for the examination of the effect of spatial pattern of disease and disease incidence on the efficiency of sampling protocols and in parameterising a general theoretical model for describing the spatio-temporal development of plant diseases.

NOTATION

General notation

γ	Correlation coefficient
τ	Holding variable used in Central Limits transformation
μ	Mean
π	Pi
σ	Standard deviation
c	Constant
esd	Applied environmental standard deviation
g	Gram(s)
I	Uniform distribution integer, 1-100 inclusive
mm	Millimetre(s)
ms^{-1}, m/s	Metres per second
n	Integer value within a series
n	Sample size
p	Probability
R	Random deviate (unspecified distribution)
t	t-value from t-test
t	Time. If subscripted, time measured in generations
U_1, U_2	Uniform distribution random deviates
X_1, X_2	Normally distributed random deviates

Genetics-related notation

η	Haploid genomic complement
A	Ploidy
A_1, A_2, A_3	Alleles 1, 2 and 3
E	Environment
G	Genotype
h	Degree of selective disadvantage
h^2	Realised heritability
H	Population heterozygosity
i	Selection intensity
L	Number of loci defining character
m	Migrant frequency

N	Population size
Ne	Effective population size
p, q	Frequency of selectively favoured and selectively disadvantages alleles
P	Phenotype
P_S	Selection pressure
R	Response to selection
s	Coefficient of selection
S	Selection differential
u, v	Forward (A_1 to A_2) and back mutation rates
V_E, V_G, V_P	Environmental, genotypic and phenotypic variances

Epidemiology-related notation

θ	Aggregation parameter
λ	Number of infections
D	Distance
e	Residual error term
H	Host
i, j	Positional integers (descriptive)
m	Epidemic shape parameter
P	Pathogen
P_g	Pathogen activity
r	Rate
R^2	Coefficient of determination
T	Temperature
y	Amount/level of disease

I. General Introduction

MODELS AND MODELLING

When considering models of any description, it is important to understand exactly what constitutes a model. A **model** is merely a representation, usually mathematical, logical or iconic, of a system or series of related events. Diagrams, mathematical formulae and computer simulations are varied, but valid examples of models.

Model classification

Given that anything from mathematical formulae, to diagrams, to a three dimensional physical structure comprised of short rods and coloured beads, can all be described as "models", it is vital that the word "model" is defined clearly and unambiguously. Unless stated to the contrary, "model" is taken here to mean a computer simulation program, and the modelling techniques employed are those of computer programming and software engineering.

Consideration of a model is meaningless without reference to the system being modelled . A **system** is defined as a set of entities that act and interact together. In practice, what is meant by "the system" will depend on the objectives of a particular study (Law & Kelton, 1991). Often, the boundaries of the system and of the model are arbitrarily defined, and inevitably the model is better defined than the system (Bratley *et al.*, 1987). The majority of interacting systems in the real world are too complicated to be comprehensively understood, and hence cannot be modelled in their entirety. This leads to the necessary compromise of modelling only the most important components of the system. How well a modelling objective is achieved will depend on both the state of knowledge about the system to be modelled, and how well the modelling is done.

Models can be classified by their structure, their function and/or their intended application. Commonly, models are classified according to their operational and organisational characteristics (functional classification). Operational classification places models into one of two broad categories: **deterministic** models predict a single, representative or "average" outcome, whereas **stochastic** models are of a more statistical nature, employing probability functions to give a distribution of possible outcomes.

Organisational classification describes models as mechanistic or empirical according to the following regime: **mechanistic** models take account of the mechanisms through which change occurs, using theoretical or experimentally derived information to describe

processes in a hierarchical manner, with larger processes being described by their component processes; **empirical** models, conversely, take no account of the mechanism by which change occurs. Empirical models (sometimes referred to as "equation-based" models) process data in a manner which may bear no direct relation to the processes within the system (Anderson, 1991). Emphasis is placed exclusively upon the net effect of component processes.

As stated previously, this study is concerned with computer-based simulation modelling. "**Simulation**" is a generic term describing many different activities. When used in a scientific context it normally refers to the construction of an abstract model representing a real system. The simulation describes the pertinent aspects of the system as a collection of equations and/or mechanisms, commonly realised in the form of a computer program.

Simulation models are generally classified with respect to an independent variable (e.g. time). **Continuous simulation** is appropriate for systems that vary constantly with time, and continuous simulation models will generally centre on the solving of differential equations. If, however, the system is comprised of a series of events, with events happening at finite time points (not necessarily evenly spaced) **discrete-event simulation** (sometimes called discontinuous or digital simulation) would be appropriate. The techniques employed in either simulation methodology can be quite dissimilar. The situation is further complicated by these definitions representing extremes of a range containing hybrid systems, where, for example, a discrete event might cause change in the value of a continuous state variable, or a continuous state variable, upon attainment of a threshold value, causes a discrete event to occur (Law & Kelton, 1991).

It must be appreciated that any model is unlikely to conform exactly to any one of the above descriptions. Nevertheless, this general description of model nomenclature is useful when referencing modelling literature, and can be an asset to the understanding of model function and application.

Modelling objectives and applications

Why model? Davies & O'Keefe (1989) identified three reasons for adopting a modelling approach:

1. The system does not yet exist, e.g. planning output from a proposed new production facility.

2. Experimentation with the system is impractical or expensive, e.g. whether it is cost-effective to invest in new machinery.

3. Experimentation with the system is inappropriate or dangerous, e.g. planning the movements of emergency services in the event of a forest fire.

Wagenet (1991) categorised model usage into three fields: research, management and education. The same author attributed the recent pre-eminence of simulation modelling to the rapid advance in affordable microcomputer technology, occurring coincidentally with the retirement of a large number of experienced field scientists to be replaced by inexperienced, but mathematically adept young professionals. Whatever the reasons or benefits, simulation modelling is now an established scientific discipline.

In very general terms, models can be used for three purposes: (1) **investigation**, to enhance the understanding of the behaviour of a system, through simplification of the system into easily understandable functional components; (2) **comparison**, to assess the effects on the system of changing key variables; (3) **prediction**, to determine the state of a system at some future point in time, to a degree of accuracy defined as being sufficient by the model user. This division of model function into "investigative", "comparative" (which together can be defined as "descriptive") and "predictive" models, despite being somewhat inelegant, is nonetheless useful. It should be pointed out, however, that the division between description and prediction is ill-defined, and will shift in response to changing assumptions and expectations on the part of the user of the model. In this respect, the expectations of the user are of vital significance. False expectations, invalid assumptions and misconceptions on the part of the user will render the best of models useless for a given application.

Modelling methodology

The modelling exercise can be partitioned into three categories: building, studying and testing (Elston, 1989). Modelling projects rarely progress smoothly through these phases, and defects or deficiencies discovered during the study or test phases result in a return to the building phase. Changes to the model would then necessitate a repeat of study and testing (Figure 1.1).

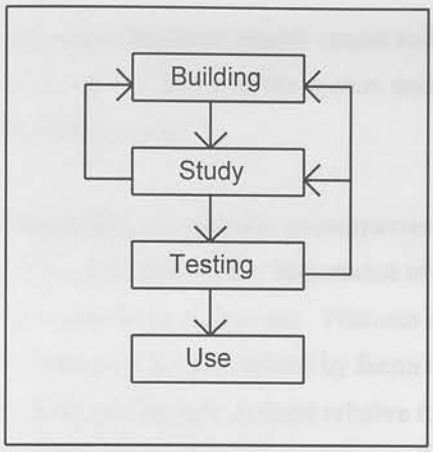


Figure 1.1. Diagrammatic illustration of the relationships between the identifiable phases of the modelling exercise (Elston, 1989).

In the building of the model, alternatively referred to as the design and development of the model, the level of detail included depends on both the intended purpose for the model, and on the time and resources available for modelling. The model is based upon the assumptions made by the modeller, which in turn are likely to be based on a partial understanding of the system being modelled. Future study of the system treats these assumptions as facts, and hence the results of such study are only as valid as the underlying assumptions. Assumptions used in model formulation, design and development must, therefore, be stated in the description of the model algorithm(s).

Qualitative description of model behaviour answers the question "how", whereas quantitative description answers the question "how much". The qualitative behaviour of stochastic models, by definition, will show more diversity than that of corresponding deterministic models. With stochastic models it is therefore important to describe not only the mean (or typical) behaviour of the system, but also provide a description (e.g. standard error of the mean) of the range of behaviours.

Model testing is commonly referred to as validation and verification. Verification is taken to mean checking that the model operates true to its design, and that (in the case of a computer-based simulation model) the program is free from coding or semantic errors. Such checks are rarely exhaustive (Bratley *et al.*, 1987) as verification errors are likely to become apparent during the course of validation procedures. Validation establishes that the model, correctly implemented, provides a satisfactory approximation to reality, and should provide an assessment of the significance of disparities. In this instance, "satisfactory" will be defined relative to the objectives of the modelling exercise. Reasons for imperfect

agreement between model output and observations from the modelled system include inherent variability in the system and its environment, and the effect of factors excluded from the model.

Modelling, the activity, encompasses more than just modelling, the techniques. Wilson (1991) highlighted the importance of essentially non-technical considerations to the success of a modelling endeavour. Features contributing to the overall worth and quality of a model (which may be determined by factors such as its suitability and ease of use, accuracy of its output, and benefit defined relative to the specific modelling objectives) include availability, maintenance and support, an accessible structure and the consequent opportunity for independent appraisal, and documentation. The most sophisticated model in the world is unlikely to receive widespread use and acceptance if the documentation is unhelpful, or worse, misleading. Wagenet (1991) highlighted the potential benefits of adopting a team approach to the development of models, with team members drawn from different backgrounds bringing different skills and abilities into the modelling project. Positive consequences can be seen both in the quality and comprehensive nature of the models resulting from such collaborations, and in the development of the members of the modelling team as each learns techniques and principles from the other members.

Ultimately, a decision must be taken to stop modelling. Modelling is most profitable when the interactions between components of a system have been characterised, but the system has yet to be considered as a coherent unit. It is time to stop modelling when the goals of the modelling exercise have been achieved, or when little further benefit can be gained. This decision can have its basis in science, in funding or in time limits.

COMPUTER PROGRAMMING LANGUAGES

A simulation program is a special kind of computer program because it attempts to mimic reality at some level of detail and approximation. This has implications for the programming techniques used to implement the model. An underlying principle of simulation modelling is that the structure of the model should directly represent the structure of the phenomenon or process being modelled (Rossiter, 1991).

The computer programming language forms the interface between the programmer and the machine, the mechanism whereby the instructions of the programmer are translated into a format that the machine can understand and act upon. Programming languages can be classified with reference to a four-level hierarchy (Figure 1.2).

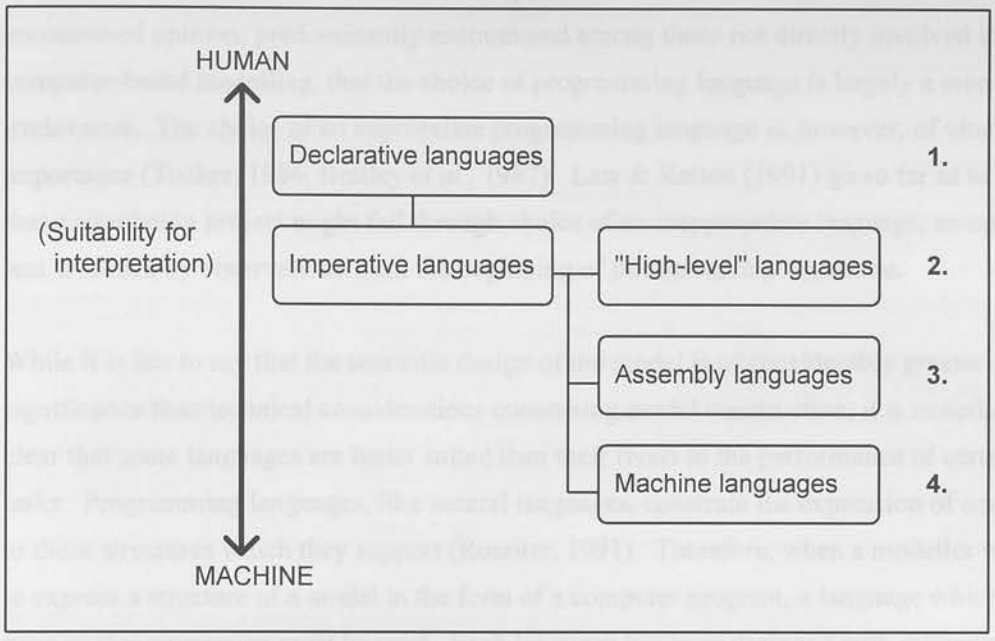


Figure 1.2. Generalised classification of computer programming languages.

Languages are either declarative or imperative. In physical detail, declarative languages most resemble natural languages, and are thus most intuitively understood by humans. Imperative languages are more abstract in description, and at the very lowest level (machine code) take the form of symbolic representation of machine commands; very efficient at the machine level, but at the expense of ready comprehension by humans (Tucker, 1986). The principal difference between declarative and imperative languages is as follows: when using declarative languages, the problem is defined and the system (language + machine) charged with finding the solution; with an imperative language, the method by which to obtain the solution must be given. Declarative languages therefore place greater demands on system resources by being a further stage removed from machine-ready format.

An additional level of language classification concerns the function of the interface between language and machine. Languages can operate using either compiled code or interpreted code. Compiled languages involve an additional stage ("compilation") before a program is in machine-ready format. Interpreted languages are directly "interpreted" at run-time. The additional stage of compilation places a burden on the programmer, whereas an interpreted language, by failing to present the program in a machine-friendly format, places a burden on the system. Low-level machine languages, by definition, require no intermediate preparation.

In the development and construction of computer-based simulation models, it is a widely encountered opinion, predominantly encountered among those not directly involved in computer-based modelling, that the choice of programming language is largely a matter of irrelevance. The choice of an appropriate programming language is, however, of vital importance (Tucker, 1986; Bratley *et al.*, 1987). Law & Kelton (1991) go so far as to state that a simulation project might fail through choice of an inappropriate language, an opinion lent credence by observations from the beginning of this research programme.

While it is fair to say that the semantic design of the model is of considerably greater significance than technical considerations concerning model construction, it is nonetheless clear that some languages are better suited than their rivals to the performance of certain tasks. Programming languages, like natural languages, constrain the expression of concepts to those structures which they support (Rossiter, 1991). Therefore, when a modeller wishes to express a structure of a model in the form of a computer program, a language which permits that expression must be used. Each language has been designed with reference to a certain range of potential applications (Marcotty & Ledgard, 1987). Subtle, and sometimes not so subtle differences among programming languages mean that it is essential that an appropriate modelling medium be chosen.

There is no single "best" modelling language. Key issues in determining the best programming language for simulation modelling are versatility vs. sophistication, and attainment vs. potential. Much depends on the objectives of the modelling exercise, the nature of the system to be modelled, and on the experience, inclinations and abilities of the modeller. The first choice faced is whether to use a dedicated simulation language or a general-purpose programming language. Simulation languages allow the description of a model at a higher level than that allowed by a general-purpose procedural language. The emphasis is on describing the mathematical structure of the model, rather than describing how these structures can be solved (Rossiter, 1991). The principal advantages are that the program forms a closer representation of the problem than that conferred by a general-purpose language, and that programs are consequently shorter, rapidly coded and easier to manage thereafter. Disadvantages are that models must be expressed with exclusive reference to the special-purpose features of the language, and that if the underlying structure of the modelled system does not map well to these features, the programmer will have great difficulty expressing the problem in a form acceptable to the language. Some of the principal advantages and disadvantages are described in Table 1.1.

Table 1.1. Advantages and disadvantages of simulation languages with respect to general-purpose programming languages (adapted from Shannon, 1975).

	Advantages	Disadvantages
General-purpose languages	(1) Few restrictions imposed on format of output (2) Common and widespread usage, and support facilities	(1) Simulation structures require longer programming time (2) Debugging of simulation terms not available
Simulation languages	(1) Simulation structures require less programming time (2) Superior error-checking mechanisms (3) Direct expression of simulation concepts (4) Automatic generation of data and distributions used in simulations	(1) Rigid definition of output format (2) Reduced flexibility (3) Increased demand on system resources

A selection of programming languages were considered for this project:

Fortran-77: Fortran-77 (ANSI, 1978) is a general purpose numerical computation language, used widely within the academic scientific community. The language is well standardised, and is highly portable across a wide range of computer processors. The widespread use of Fortran, and the consequently ready access to technical support and advice, is a major positive attribute in favour of the adoption and use of Fortran. The language is, however, rather clumsy and less than user-friendly for those unversed in its use. Fortran was conceived in 1954, and deficiencies in the language arise from unfortunate decisions taken in the 1950's, which have been perpetuated to ensure upward compatibility from earlier Fortran dialects (Rossiter, 1991). There is no exception-handling capability, and dynamic (unformatted) variables are not supported (Fortran requires that the number and size of variables be known at compilation) which is highly restrictive. The continued use of Fortran is arguably a historical artefact: many experienced modellers, being well practised in its use, continue to use it rather than "wasting" time learning a new language.

Derivatives of Fortran designed specifically for simulation use are available. Examples include GASP-II (Pritsker & Krivat, 1969), GASP-IV (Pritsker, 1974) and SLAM (Pritsker, 1986). All are recognisably Fortran descendants, but feature advanced features unavailable in standard Fortran. In adopting one of these extended Fortran-based compilers, however, the benefits conferred by Fortran (widespread use, availability of technical support etc.) would be lost.

Basic: Basic was initially developed for use as a training language, and despite its many limitations, remains well suited to its intended application. Because of its accessibility and ease of use, it has been, and continues to be used in the construction of models (Hull, 1977; Murphy, 1983; Muetzelfeldt, 1991). Basic is an interpreted language. Despite its many positive attributes, the language is unsophisticated, dated and relies heavily on the GOTO statement for control of program execution. This can result in the physical structure of the computer program becoming divorced from the structure of the underlying computation.

Pascal: Pascal (Jensen & Wirth, 1974), a direct descendent of the ALGOL-60 programming language (Naur, 1963), started life as a language for teaching algorithm design and structured programming methodology, and can be viewed as the successor to Basic for the role of general introductory training language. Perhaps as a consequence of this educational role, the language is used widely within the academic community as a general purpose, high-level programming language. Modelling work of direct relevance to this study has been conducted using Pascal (Open University, 1987). The language is characterised by being highly structured, promoting precise expression of structure to the point of actively disallowing common "poor programming" practices (Davies & O'Keefe, 1989) and hence prevents many common semantic and syntactical coding errors. There is an internationally recognised standard (ANSI, 1983) which promotes portability across computer platforms.

To its detriment, the strong structuring and typing that result in a highly precise language can restrict its versatility, preventing the efficient (and hence "quick") coding of essentially simple structures. A major disadvantage concerns limitations with respect to array arithmetic and handling (Tucker, 1986).

Prolog: (Kowalski, 1981; Clocksin & Mellish, 1981) A "logic-based" family of declarative languages designed for the construction of programs requiring the manipulation of logical data, rather than numerical computation. *Poplog* and *Pop-11* (generic Prolog languages), designed and marketed jointly by the Universities of Edinburgh and Sussex, were considered for use in this study, largely because of the ready access to technical advice and support. Prolog has been accused of "*linguistic eccentricity*" (Rossiter, 1991). In addition, the models under consideration require a high degree of numerical computation, and Prolog, despite being widely used for simulation purposes, was therefore rejected as being unsuitable.

C: (Kernighan & Ritchie, 1978) C is the language currently in vogue for the development of systems-type applications. C is a highly versatile, and widely used general purpose

development language, but is renowned for having a long learning-curve and being difficult to use (Rossiter, 1991). The language was developed for writing machine operating systems, requiring low-level access to data and precise control over compilation. It was designed to encourage economy of expression (to maximise operating efficiency), and as such appears (superficially) biased towards the consideration of the machine rather than the programmer. C features an extensive function library which enhances its versatility, but again, makes it a difficult language to master. As with all programming languages, precision and versatility can be in opposition, with one being achieved at the expense of the other. C was rejected for this project by virtue of its reported difficulty of use.

Simula: Simula (Dahl *et al.*, 1967), a derivative of ALGOL-60 (Naur, 1963), is a specialised simulation language developed at the Norwegian Computing Centre, Oslo. The language is an example of an Object-Oriented programming language. The concept of Object-Oriented Programming (OOP) allows groups of variables to be handled collectively, i.e. as an object (Birtwistle, 1973; Pooley, 1987). Groupings of like objects become a class. This hierarchical organisation into objects and classes has positive consequences for the structuring and handling of data, and to a certain extent permits the disassociation of program control from program execution (Kerr *et al.*, 1990). This can enhance the efficiency of program execution. Simula is a complicated language (Bratley *et al.*, 1987) with tremendous, but rarely tapped potential. Documentation is of variable quality and comprehension, and the language is characterised by enthusiastic but minority use.

C++: C++ (Lippman, 1989; Ellis & Stroustrup, 1990) is a derivative of C, and a direct descendant of Simula. Its design was contributed to by original members of the Simula design team, and the language contains many features currently lacking in Simula. C++ applications are portable to any machine with a C compiler. The language was rejected, however, for essentially the same reasons as C. In addition, use of C++ is not widespread, and technical assistance would have been very limited. The increasing adoption of C++ as a specialist simulation language (Bujakiewicz & van den Bosch, 1991) has meant that some of the above considerations are becoming less valid.

A humorous, but nonetheless highly illustrative description of task performance by commonly used high-level programming languages, contributed anonymously to the British Computer Association for the Blind (BCAB) bulletin board service, is presented as Appendix A1.

The decision was taken initially to use the simulation language PC-Simula (Simula a.s.). It later became apparent that the language was overly powerful, complicated and difficult to use for the envisaged modelling exercises, and is poorly suited to discrete-event simulation. Modelling activities were transferred to Turbo Pascal (Borland International Inc.), version 6 for MS-DOS. Turbo Pascal is a general purpose structured procedural language. Although there are products better suited to the task of computer-based simulation modelling, Turbo Pascal was chosen because of its versatility, comparative ease of use, and its widespread use within the scientific community. While unable to offer specialised simulation features and functions, using a common general purpose computer language to develop simulation models offers crucial benefits of a predominantly non-technical nature. The relative abundance of peers has positive consequences concerning the pooling and exchange of ideas and experience. Technical support and advice concerning common programming problems tends to be personal, informal, generally verbal, and above all rapidly and readily available. All these factors contribute favourably to the general "worth" of a given language. Exposure to ideas, techniques and technical "fixes" can outweigh the advantages conferred by a more sophisticated programming language.

MODELLING MECHANISMS OF CHANGE IN CROP POPULATIONS

This research programme centres on the design, development, implementation and validation of computer-based simulation models, modelling changes in crop populations brought about by (a) evolutionary forces, and (b) the activity of pathogenic agents. The models presented have been developed as mechanistic, stochastic simulations, modelling the underlying mechanisms of change rather than empirically mimicking experimental observations or results. Modular (procedure-based) programming has been implemented throughout, to ensure that the structure of the models produced would be accessible to biological scientists with only limited computer programming skills, enabling them to make amendments and improvements to the models if and where they see fit. In taking this approach it is intended that the models be open to independent appraisal, stimulate discussion and consideration of the mechanisms modelled, and be suitable as platforms for continued development.

The initial aim of the research programme was to design and develop a computer-based simulation model of selection in crops, for use as a decision-support tool in plant breeding. As the research evolved, the objectives became amended and revised, increasingly centring on the development of a modelling methodology and subsequent application of that methodology to the simulation of mechanisms acting within and upon plant populations.

The models so developed then had to be verified for conformity to the established methodology, and validated against experimental data and accepted mathematical descriptions of the subject biological mechanisms.

II. PSELECT: Computer Simulation of Selection in a Hypothetical Crop Species.

INTRODUCTION

Principles of evolutionary genetics

The single most significant obstacle to the understanding of this field of biology, is one of definition. The science of population genetics can best be viewed as a fusion of scientific disciplines, and one consequence of this has been the evolution of definitions rather than their rigid specification. A common occurence is where two scientists will use the same description for different phenomena. For example, the term "gene frequency" is used to denote the frequency of a gene (specifically the allelic composition of that locus or group of loci) in a population, or the frequency of a component allele. These are subtly different properties, but are both invariably referred to as "gene frequency". An understanding of the context in which descriptive terms are used is necessary.

The gene is now established as a workable scientific model, and a discussion of the gene as the base unit of inheritance is not required. It is sufficient in this context to state that the genotype of an individual is described by a large number of genes, each of which is likely to have more than one allele segregating within the population, the nature and combinations of which specify the genotype. The population consists of a number of individuals. The population can therefore be described by the occurence of alleles within it, i.e. by allelic frequencies. The frequency of alleles in a population will be subject to change as a consequence of a range of factors; referred to as evolutionary forces. Major evolutionary forces that can be considered are selection, recurrent and non-recurrent mutation, migration, and dispersive (random) processes.

Fundamental processes

(1a) Natural selection

Selection acts via disproportionate contribution of viable progeny to the following generation by superior individuals in the parental generation. This contribution describes an individual's fitness with respect to selection. The two components of fitness are therefore viability and fertility. If differences in viability or fertility between individuals are in any way associated with the presence, form or absence of a gene, then selection can be expected to act upon that gene. Selection affects the gene in question by selectively eliminating genotypes exhibiting inferior forms of the gene. The frequency of an allele in the offspring will therefore be different from that in the parents.

Selection can only effect change on populations showing phenotypic variation for a character (Robertson, 1960), and acts indirectly upon the genotype through differences in fitness among phenotypes. The strength of selection is the proportionate reduction in contribution of a particular genotype with respect to a standard genotype, defined as having a fitness of 1. This quantity is called the coefficient of selection, *s*. The contribution of the less favoured genotype is 1-*s*.

The mathematical treatment of selection is complicated, involving treatment of the dominance relationships at the locus in question and environmental considerations with respect to the phenotype (Lewontin, 1974). Change in allelic frequencies as a consequence of selection may, however, be summarised according to the following formula:

$$dq = -sp_0q_0\left[q + h(p_0 - q_0)\right] \Big/ 1 - 2(hsp_0q_0) - s(q_0)^2$$

where p_0 and q_0 are the initial frequencies of the alleles, s is the strength of selection, and h is the degree of selective disadvantage of the heterozygote relative to the unfavoured homozygote. For example, if the A_1A_1 homozygote has a fitness of 1, and the A_2A_2 homozygote has a fitness of 1-*s*, the heterozygote A_1A_2 has a fitness of 1-h*s*. If $s = 0.2$, and A_1A_2 is of exactly intermediate fitness to A_1A_1 and A_2A_2 (the condition of additive allelic dominance), then h = 0.5. The relative fitness of A_1A_1, A_1A_2 and A_2A_2 are therefore 1, 0.9 and 0.8 respectively.

The new allelic frequency after one generation of selection, q_1, is given by the formula:

$$q_1 = q_0 - (hsp_0q_0) - s(q_0)^2 \Big/ 1 - 2(hsp_0q_0) - s(q_0)^2$$

It can be seen that the effect of selection on allelic frequencies depends on the allelic frequencies in the parental generation, the dominance relationships among alleles at the locus in question, as well as the degree of selective advantage or disadvantage conferred by the presence of the allele under consideration. The factors affecting natural selection are coverered by Lewontin, (1974), and Falconer, (1981).

(1b) Applied ("artificial") selection

It is with the action of artificial selection that this study is primarily concerned. The processes involved in applied selection are essentially the same as those discussed above, with the principal difference being that fitness is artificially defined by the experimenter, and hence genetic change is towards a pre-specified objective or objectives. Undesirable phenotypes are lethal as the experimenter does not breed from these phenotypes, which consequently have no contribution to the following generation.

When considering segregation at a single locus, response is maintained by repeatedly selecting against unwanted genotypes. This process is continued until the unwanted allele is no longer segregating within the population. Expression of the unwanted allele is effectively lethal, $s = 1$. The majority of characters subjected to applied selection are, however, polygenic. Response in the desired direction is achieved by selecting for those phenotypes within the population exhibiting the greatest degree of conformity to the desired phenotype. For example, if an increase in weight is desired, the heaviest individuals are chosen to be the parents for the following generation.

The response to selection, R, the difference between the mean value of the character in the parental and offspring generations, depends on the phenotypic variation within the population, the selection pressure applied to the population, and the heritability of the character:

$$R = ih^2\sigma_P$$

where σ_P is the phenotypic standard deviation in the population, i is the intensity of selection, the difference between the mean phenotypic value for the whole of the parental generation and that of the group of individuals selected to be parents (the selection differential, S) as a proportion of the phenotypic standard deviation in the population, S/σ_P. h^2 is the heritability, a measure of the relative importance of heredity in determinig the phenotypic value of the character upon which selection is applied (Wright, 1921).

(2a) Recurrent mutation

Mutation is the ultimate source of genetic variation. If allele A_1 mutates to allele A_2 at a charateristic rate, and the reverse occurs at a different characteristic rate, it is apparent that the alleles will eventually stabilise at an equilibrium frequency in the population. If the rate

of forward mutation (from A_1 to A_2) is u, and the rate of back mutation is v, the net change in allelic frequency at a locus attributable to recurrent mutational events can be described by the formula:

$$dq = up_0 - vq_0$$

where p_0 and q_0 are the initial frequencies of A_1 and A_2 respectively.

The degree of change is therefore dependent not only on the rate of mutation from one allele to the other, but also on the frequency at which that allele is present in the population. By equating dq to 0, the equilibrium frequencies may be found:

$$q = u/(u+v)$$

Mutation rates are invariably low; commonly between 10^{-5} and 10^{-6} per generation per locus (Haldane, 1949; Sprague & Schuler, 1961; Falconer, 1981). It is observed that the rate of mutation from the selectively wild-type allele to the mutant form is generally of an order of magnitude greater than the reverse event (Wright, 1951; Schlager & Dickie, 1967). One would therefore expect the mutant form to become stabilised in the population at a frequency of 0.9 if mutation were the only evolutionary force acting upon the population. This is clearly not the case.

In the absence of mutagenic agents recurrent mutation is not a powerful evolutionary force. Changes in allele frequency attributable to mutation alone are slow. Mutation can become an important consideration where the population size is small, but in an infinitely large population the importance of mutation is primarily as a source of polymorphism upon which selection can act.

(2b) Non-recurrent mutation

Unique, or non-recurrent mutations are of little importance as an evolutionary force (Kimura & Ohta, 1971). Due to random processes (described in a later section), the probability of a unique mutation being retained in the population is very small. Additionally, for all organisms other than haploids, the initial mutation would be expected to be present in the heterozygous condition, and therefore stands no more than a 0.5 probability of representation in the next generation.

Non-recurrent mutations can become significant in small populations, where fixation of the new allele in the population becomes a significant possibility.

(3) Migration

The genetic composition of the population changes if the physical nature of the population is altered. Migration, the introduction of new individuals into the population, will add new alleles to the population, and hence will change the relative frequencies of alleles within that population. The change in allelic frequency, dq, due to one generation of migration is summarised by the formula:

$$\mathrm{d}q = \mathrm{m}(q_m - q_0)$$

where m is the proportion of the population comprised by new immigrants, q_m is the frequency of the allele in the immigrant contingent, and q_0 the allelic frequency in the native population. It can be seen that the most important factors affecting allelic frequencies are the rate of immigration and the difference in allelic frequency between the native and immigrant populations.

(4) Dispersive processes

The aforementioned agents of genetic change are predictable in magnitude and direction, and are referred to as systematic processes. Dispersive processes differ from systematic processes in that they are predictable in magnitude only. Dispersive processes arise as a result of finite population size. In a small population, the assumption that in the absence of the systematic evolutionary processes gene frequencies will remain stable from generation to generation is incorrect. Gene frequencies are subject to fluctuation between generations as a consequence of sampling variance: the gametes that carry genes to the next generation are a sample of those present in the parental generation. The effect of sampling variance can be summarised by the formula:

$$V_{\mathrm{d}q} = {(p_0 q_0)}/{2N}$$

where $V_{\mathrm{d}q}$ is the variance of changes in gene frequencies per generation as a result of sampling, p_0 and q_0 are the initial allelic frequencies, and N is the population size. As the

starting point for this process is different with each successive generation, the effects will be cumulative (Buri, 1956).

The consequences of the dispersive processes are seen as random drift, inter-subpopulation diversity, intra-subpopulation uniformity, and increased homozygosity. The random fluctuation in allelic frequencies within a given subpopulation that has arisen as a consequence of sampling effects inevitably leads to differentiation of subpopulations within the general population (Rich *et al.*, 1979). The variance among lines is given by the formula:

$$V_q = p_0 q_0 \left[1 - \left(1 - \left(1/2N \right) \right)^t \right]$$

where V_q is the variance among gene frequencies, p_0 and q_0 are the frequencies in the initial population, N is the population size, and t is the number of generations from the initial generation, 0. If a single subpopulation is considered, an inevitable consequence of fluctuating allelic frequencies is the loss or fixation of alleles segregating within the population. This process is called random drift.

Phenotypic truncation selection

Phenotypic truncation selection is the selection protocol adopted throughout this study. Truncation selection identifies and isolates individuals in order of phenotypic merit, and like all selection mechanisms is therefore reliant upon phenotypic variation within the population. If a population displaying a normally distributed range of phenotypes is considered, truncation selection can be described with reference to Figure 2.1. The selection differential, S, per generation is given by the difference between the population mean (μ_P) and that of the selected proportion of the population (μ_S). Selection differential is therefore a feature of both the phenotypic variance within the population, and the number of individuals selected from the tail of the distribution. If the heritability of a character, h^2, is known, the response to selection can be predicted by the product of S and h^2. Conversely, by measuring response to selection, the same relationship can be used to estimate the heritability of the character under selection (realised heritability). The selection differential, expressed as a proportion of the phenotypic standard deviation (σ_P), gives the intensity of selection (i). It is these relationships that will be considered in the validation and verification of the PSELECT model against observed selection responses.

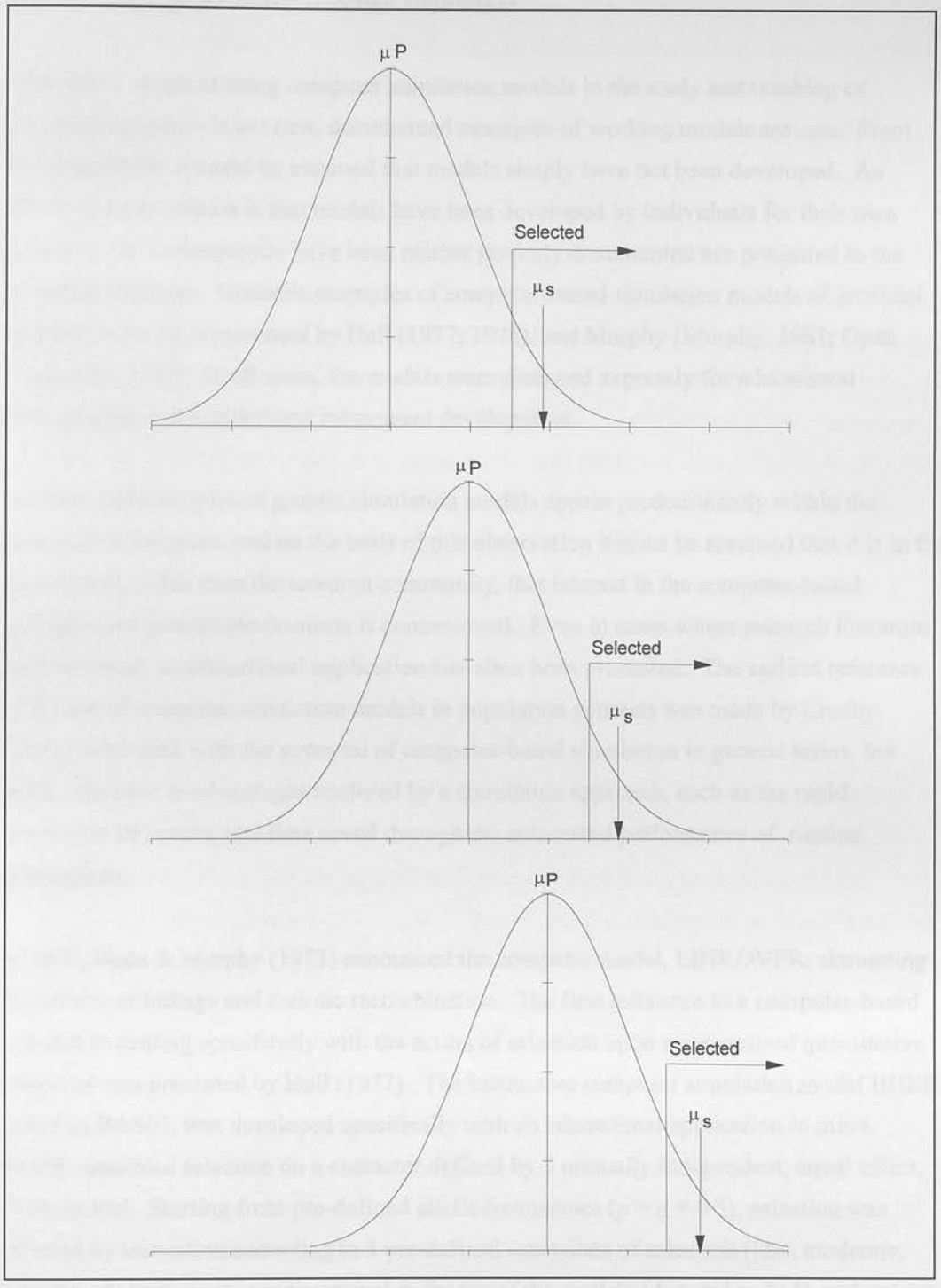


Figure 2.1. Diagrammatic representation of truncation selection in favour of increased phenotypic score (denoted by shift in population mean, μ_P, to the right) by selection from the right-hand tail of the phenotypic distribution. Subtracting the population mean from the the mean of the selected proportion of the population (μ_S) gives the selection differential, *S*.

History of computer-based selection simulation

While the concept of using computer simulation models in the study and teaching of population genetics is not new, documented examples of working models are rare. From this observation it could be assumed that models simply have not been developed. An alternative explanation is that models have been developed by individuals for their own purposes, and consequently have been neither properly documented nor presented in the scientific literature. Noteable examples of computer-based simulation models of artificial selection have been presented by Hull (1977; 1978), and Murphy (Murphy, 1983; Open University, 1987). In all cases, the models were designed expressly for educational purposes, and none underwent subsequent development.

Documented examples of genetic simulation models appear predominantly within the educational literature, and on the basis of this observation it must be assumed that it is in the educational, rather than the research community, that interest in the computer-based simulation of genetic mechanisms is concentrated. Even in cases where research literature has been used, an educational application has often been presented. The earliest reference to the use of computer simulation models in population genetics was made by Crosby (1961), who dealt with the potential of computer-based simulation in general terms, but made reference to advantages confered by a simulation approach, such as the rapid generation of results, and time saved through the automated performance of routine calculations.

In 1973, Dean & Murphy (1973) announced the computer model, LINKOVER, simulating the effects of linkage and meiotic recombination. The first reference to a computer-based simulation dealing specifically with the action of selection upon a generalised quantitative character was presented by Hull (1977). The interactive computer simulation model BHER, coded in BASIC, was developed specifically with an educational application in mind. BHER simulated selection on a character defined by 5 mutually independent, equal effect, diallelic loci. Starting from pre-defined allelic frequencies ($p = q = 0.5$), selection was effected by truncation according to 3 pre-defined intensities of selection (low, moderate, intense). Selection was unidirectional in favour of the p allele (denoted as "+"), and output to screen listed response to selection in terms of the frequency of this allele, f(p). The number selected per generation remained constant, leaving the intensity of selection to be specified by varying the number of offspring produced per generation. The model could not simulate allelic dominance, but was able to address the contribution of environment to phenotype, through the specification of fixed values (high or low) for heritability of the

character. Hull made a second significant contribution to genetic modelling with the SURF-6 simulation model (Hull, 1978). SURF-6, simulated the construction of adaptive surfaces (Wright, 1932) through the action of selection in natural populations. This model undoutedly attained a greater sophistication than BHER, but appears to have found less application.

The simulation model WHEAT (Murphy, 1983), like the earlier BHER model (Hull, 1977) simulated the action of selection of a 5-loci metric character, this time specifically coding for plant height in a simulated wheat crop. All loci were mutually independent (no linkage or epistatic effects), diallelic, and of equal genotypic contribution. As an improvement over BHER, WHEAT was able to simulate full allelic dominance as well as additive dominance. A feature of the WHEAT model was the presentation of printed output, displaying line plots of various lengths representing plant heights within the simulated population. This graphical, and highly visual ouput significantly contributed to the educational potential of the model. A significant disadvantage of the model was that it was not interactive, with changes to the program code being required to specify changed parameter values. WHEAT was coded in BASIC for execution on a DECsystem-20 minicomputer (Digital Equipment Corp.).

WHEAT was followed by EVOLVE (Murphy, 1984), developed to model natural selection and the evolutionary implications of merging two genetically divergent populations. Pure-breeding representatives of both populations had a selective advantage over hybrids resulting from inter-population crosses, with the relative disadvantage of hybrids being defined by the ratio of genotypic contribution from each population (fitness is minimal for the 1:1 hybrid). In its original form, EVOLVE was non-interactive. The model was, however, stochastic, and a given run of the simulation could be expected to yield a unique output. Extensions to the model allowed for the user-specification of, among other parameters, the relative fitness of pure-breeding and hybrid plants. The same author subsequently described a computer-based model simulating heritability (Murphy, 1986), and illustrated by a series of simple sensitivity analyses conducted on the parameters affecting the heritability of a metric character. This work was especially interesting for the general discussion in which the author questions the nature of computer-based simulation models, and introduces the (unnamed) concept of mechanistic modelling, in contrast to the accepted and popular empirical (algebraic) approach.

A final example of computer-based selection models is the BREED1 model (Open University, 1987), developed with reference to an earlier analogue simulation model

(Simmonds, 1979). BREED1 simulated selection in a hypothetical diploid plant species, propagated by inbreeding. Closely related finished-line parental material was crossed. Parents were defined as complementary homozygotes for the character under selection, giving rise to a homogenous F_1 population, homozygous for genomic background, but completely heterozygous at all loci defining the character under selection. The F_1 population was taken as the starting point for selection. Individual allelic scores were invisible, scores being assigned to allelic pairings. Population size was fixed at 100 and selection pressure was constant at 90% per generation. The population was described by a serial array of 1000 allelic pairings. For the F_2 and subsequent generations, a uniform distribution (0 to 1) random deviate, z, was generated for each locus. If $z < 0.25$ or $z > 0.75$, the locus became homozygous for A_1 or A_2 respectively. If $0.25 < z < 0.75$, the locus remained heterozygous. Genotypes and phenotypes were calculated from the sum of locus values and the genomic background, and population statistics were written to an output file after each round of selection. Phenotype was specified by the addition of a randomly-specified environmental component. Upwards phenotypic truncation selection was effected by generating the following generation from the highest scoring 10 phenotypes. Each selected plant contributed 10 offspring to the subsequent generation. The BREED1 model was, again, not interactive, with key parameters specified within the program code. The model was stochastic by virtue of the randomly specified environmental variability

BREED1 was used as an educational model to investigate the effects of selection upon key genetic parameters (phenotypic and genotypic response, variance components and heritability). The model was suited to this purpose by being computationally simple, and by presenting clear, visual output, easily interpreted by the user. The program code, however, was comparatively unstructured (coded as a single block with no sub-division into autonomous procedures), and physical recoding would be required to specify alternate values for key selection parameters, such as degree of environmental variation, intensity of selection and population size. The model simulated selection only in inbreeders, although facility was provided to allow single generations of outcrossing. By extension, by specifying each generation as a generation of outcrossing, it would be possible to simulate an outcrossing plant species, but this would be a far from ideal solution. BREED1 was perfectly suited to its intended application, but due to its structural limitations, further development would have proved difficult.

OBJECTIVES

The computer-based simulation model PSELECT was developed to continue the earlier works of Simmonds (1979) and Murphy (Open University, 1987). Starting from first principles, the objective was to develop a workable mechanistic simulation of selection in a crop population. The model would have to demonstrate the phenotypic and genotypic responses to directional selection upon a simplified quantitative genetic character, and would be required to simulate selection for alternative breeding regimes, modes of allelic interaction, selection pressures, population sizes and levels of environmental variation. An educational application was anticipated, but there was the firm intention of providing a platform for subsequent research application. In this respect the model would need to be stochastic, and show qualitative agreement with observed experimental plant population responses to selection.

To achieve widespread application, the model would need to be suitable for use on generally available, low-cost computer equipment, and be portable across a wide range of computer systems. In order to be accepted as a developmental platform for use in a research capacity, the physical structure of the program code would need to be sufficiently simple to make the program amenable to change by biological scientists with only moderate programming expertise, and would need to be modularised to allow for the insertion and replacement of functionally autonomous routines modelling specific mechanisms or processes.

SIMULATION

The PSELECT model has its origin in mechanical (analogue) simulation models developed and used for teaching purposes at The Edinburgh School of Agriculture (Simmonds, 1979), and was inspired in part by the BREED1 model (Open University, 1987) described previously. In Simmonds' analogue model, a simulated plant population was subjected to selection for a polygenic character, and response to selection over time in generations predicted. The model dealt exclusively with the action of selection upon the simulated population. This simplification had positive consequences for the use of the model as a demonstration or instructional device, but imposed limitations on the continued development of the model.

A diploid plant species was subjected to phenotypic truncation selection for a quantitative character. The character was defined by ten mutually independent loci, each contributing equally to the genotype of the character. Two alleles, A_1 and A_2, were present in the population. Alleles were specified by coloured beads (red for A_1, blue for A_2) with loci scored as follows: A_1A_1=3, A_1A_2=2, A_2A_2=1. Thus, simple additive dominance was simulated. Allelic frequencies within the F_1 population (fully heterozygous for the selected character) were equal, and this was modelled by having equal numbers of red and blue beads in the simulated gene pool, which was assumed to be infinite. A two-dimensional card grid simulated the population, with coloured beads being drawn at random from the gene pool, and placed sequentially within the grid to generate N(10*2) arrays, where N is the population size. Plant genotypic values were generated by the sum of allelic components, added to a constant representing the genomic background. Phenotypes were derived from the addition of random deviates simulating environmental contribution, to the genotypic scores. An illustration of this simulation protocol is presented as Figure 2.2.

Superior phenotypes were selected, and contributed an equal number of offspring to the next generation. Propagation was by inbreeding, and all genetic processes other than selection were excluded. Homozygous loci (RR or bb) were deemed fixed, taking no further part in the response to selection. Heterozygous loci were reconstituted by randomly sampling alleles (coloured beads) from the gene pool. Phenotypic and genotypic responses to selection on phenotype were monitored to a point where potential for further response to selection is limited by absence of (or a low level of) population heterozygosity (H). By simple redefinition, the basic model could simulate both dominant allelic action and over-dominance, by assigning locus values of A_1A_1=3, A_1A_2=3, A_2A_2=1, and A_1A_1=2, A_1A_2=3, A_2A_2=1 respectively.

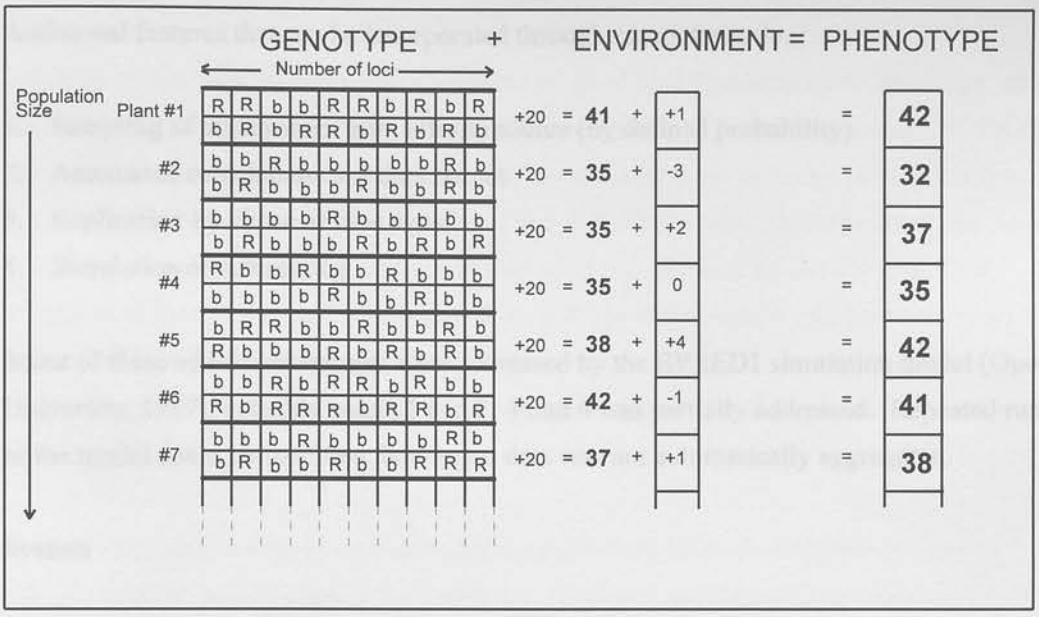


Figure 2.2. Operational protocol from analogue simulation model (Simmonds, 1979).

Key features of the analogue teaching model:

1. Illustration of the correlation between phenotypic and genotypic responses to selection
2. Stochastic by virtue of randomly specified environment
3. Demonstration of environmental masking, and the loss of favourable genotypes
4. Visual representation of increasing homozygosity and reduced potential for response to selection

Advantages:

1. Highly visual display of allelic contribution to genotype
2. Very tactile model, with a "hands-on" simulation of random sampling of alleles in segregating loci

Disadvantages:

1. All calculations manual, which is slow and gives potential for introduced human error
2. Gene pool is not infinite, causing sampling bias in favour of low-frequency alleles
3. Simulates only inbreeding (and F_1-hybrid production)
4. Replication difficult due to time constraints

Additional features that can be incorporated through computerisation:

1. Sampling of alleles from truly infinite source (by defined probability)
2. Automated calculations and data output
3. Replication by virtue of time saved
4. Simulation of outcrossing

Some of these additional features were addressed by the BREED1 simulation model (Open University, 1987), noteably points 1 and 2. Point 4 was partially addressed. Repeated runs of the model could be specified, but output data was not automatically aggregated.

System

PSELECT was developed in Turbo Pascal version 6.0 (Borland International Inc.), running under the DOS operating system (Microsoft Corporation), and compiled on an IBM PS/2 55sx personal computer (IBM Corporation) equipped with 4 megabytes of Random Access Memory (RAM). No math co-processor was fitted.

The model was tested on a range of IBM-compatible machines, and there are no unusual hardware requirements. Recommended minimum system specification is generic 80286 processor, DOS 3.3 and 512kBytes RAM. Performance will, of course, be improved by a higher specification. The program will run from floppy disk, but the resulting reduction in performance is significant, especially when high repetition is specified. It is therefore recommended that PSELECT be run from hard-disk.

Algorithm

Number of loci

A simplified polygenic character is modelled. The character is defined by a user-specified number of diallelic loci, L, with all loci having a mutually equal effect on the genotype. Spontaneous mutation is excluded, as are epistatic interactions. The genotypic contribution of a locus is defined by its allelic complement, according to the allelic values specified and the dominance relationships between alleles. L must be within the range 1 (simulating Mendelian segregation at a single locus) to 20.

Ploidy and dominance relationships

Additive allelic dominance signifies a heterozygote of exactly intermediate genotypic value with respect to the homozygous condition of its component alleles. For example, if the A_1 allele scores 5 and the A_2 allele scores 3, the diploid homozygote A_1A_1 has a genotypic value of 5+5 = 10, and A_2A_2 has a value of 3+3 = 6. Under additive dominance, the A_1A_2 heterozygote scores 5+3 = 8. Full allelic dominance assumes an equivalent genotypic contribution from A_1A_1 and A_1A_2 loci, with only the unfavoured homozygote (A_2A_2) being distinguishable. Varying degrees of dominance are possible, so that the heterozygote is not of exactly intermediate value, but is nonetheless distinguishable from either homozygous form.

PSELECT models a diploid organism (and by extension, allotetraploidy for the consideration of non-biochemical characters). Triploidy, autotetraploidy and higher levels of polyploidy are not considered. The model allows for the specification of either additive allelic action or fully dominant allelic action at all loci, with respective scoring regimes of $A_1A_1 = 1.5(A_1A_2) = 3(A_2A_2)$, and $A_1A_1 = A_1A_2 = 3(A_2A_2)$, being applied. A simple correction factor is applied to the case for additive allelic dominance to simulate degrees of partial dominance, allowing for A_1A_2 values of between $0.51(A_1A_1)$ and $0.99(A_1A_1)$. An extension to the basic model allows for the specification of overdominance, where $1.5(A_1A_1) = A_1A_2 = 3(A_2A_2)$.

By default, only two alleles are segregating within the simulated population. The model can simulate segregation for 3 or more alleles, but this requires a modification to the declared CONSTANTS region of the program code. It should be borne in mind that an arithmetic increase in the number of segrgating alleles necessitates a geometric increase in coding for allelic combinations.

Population size

Continued potential response to selection is dependent on the maintenance of genetic variation within the population. Selection from a larger population allows a greater intensity of selection (a feature of the proportion selected) to be practiced without sacrificing, to such an extent, the genetic variability (a feature of the total number of plants) within the selected population. The maximum population size that can be specified in PSELECT is 200. At a selection pressure of 95% (extreme) a population of 200 would allow for the selection of 10 plants. A population size of 10 is significant as it has been

hypothesized that in groups of less than 10 individuals, dispersive genetic processes (drift etc.) can negate selection response (Swanson *et al.*, 1974). Under experimental conditions, the maintenance of population size in excess of 200 plants may be unfeasible, but this limitation is not applicable for study by simulation methods.

Evolutionary forces

(1) Selection

Two fully complementary homozygous parents, such as would be found in finished inbred breeding lines or varieties, are mated to generate a defined heterozygous F_1 population. This F_1 population then forms the gene pool from which the subsequent generations are formed. The relative proportions of the two alleles, A_1 and A_2, in the gene pool are therefore equal. Population size, N, constant from generation to generation, is specified by the user. Individual plants consist of L allelic pairs (i.e. $2L$ alleles) which define the genotype. The population is defined by a serial array of N plants.

The genotypic value of a plant is given by the addition of a constant value, representing the genotypic contribution of the genomic background, to the sum value of the L loci defining the character. The genotypic background is given by the product of the number of loci defining the character, and the intermediate value of the two homozygous forms. Phenotypes are generated by the application to the genotypic scores of randomly generated environmental factors. In practice these are a set of normal distribution random deviates of specified standard deviation about a mean of zero, generated by transformation of system-supplied uniform distribution random deviates.

Phenotypic truncation selection is effected upon the F_1 and subsequent generations at a user-specified selection pressure. Selected phenotypes from generation F_t constitute the parental stock for generation F_{t+1}. Three alternative breeding regimes, monogamous and polygamous outcrossing, and selfing, can be simulated.

(2) Mutation

A mutated allele has a significantly better chance of survival if arising in the (haploid) gamete. In the diploid condition, the mutant allele would be present as a heterozygote, and hence have a 0.5 per generation probability of extinction. If arising in the haploid condition, the mutant can guarantee representation in the next generation. Mutations are a potential

consequence of errors in the meiotic transcription and translation mechanisms. The PSELECT (optional) mutation mechanism therefore introduces mutations only within the selected parents array. Mutations, A_1 to A_2 or A_2 to A_1, occur with user-specified probabilities. The specified probabilities are halved prior to implementation to simulate the gametic phase. Each locus within the parents array is sequentially addressed, and faces an equal (low) probability of mutating to its complementary form. New allelic forms (A_3 etc.) are not considered. By default, the mutation mechanism is disabled.

(3) Migration

Migrants can enter the population at a user-specified frequency. Like the mechanism simulating mutation, the PSELECT migration mechanism is optional. Migration is effective within the main population array, with randomly specified natives being replaced by migrant genotypes. For computational simplicity, only parental-type genotypes are considered for migration into the population. A migration event is simulated by redefining all loci of the the supplanted individual to be homozygous for the chosen parental form. Rate of migration can be defined by the number of individuals entering the population per generation, or by specifying a migration probability (which defines a mean rate of migration, and introduces a stochastic element to the mechanism). By default, the migration mechanism is disabled.

Breeding protocol

(1) Inbreeding

Pedigree selection is one of the more commonly adopted methods of crop improvement for naturally inbreeding crop species, such as bread wheat (*Triticum aestivum*) and barley (*Hordeum vulgare*), and certainly the most easily simulated breeding strategy. The parent plant self-pollinates, resulting in offspring derived exclusively from the genetic composition of the parent plant. Crops employing a degree of outbreeding in their natural reproductive strategy, for example rape (*Brassica napus*) and tobacco (*Nicotiana* spp.), would normally be bred by pedigree methods as inbred pure lines. Similarly, inbred lines for hybrid varieties may also be produced by forced inbreeding of naturally cross-pollinating species. An important example of this is maize (*Zea mays*).

In the model, selected genotypes progress from one generation to the next, with heterozygous loci (A_1A_2) being reconstituted from alleles drawn randomly from the gene

pool. For each segregating locus, a uniform distribution random integer, *I*, between 1 and 100 inclusive, is generated. If *I* is less than or equal to 25, or greater than 75, the locus becomes homozygous for A_1 or A_2 respectively. If *I* is greater than 25 but less than or equal to 75 the locus remains heterozygous. This is functionally analogous to the *z*-probability mechanism introduced by Murphy in the BREED1 model (Open University, 1987). Upon becoming homozygous, a locus will not re-segregate, unless subjected to a rare mutation event (which would only occur if the PSELECT mutation mechanism were enabled). The expected probability of any given locus becoming homozygous, and hence taking no further part in genotypic improvement, is 0.5 per generation. Thus the expected level of population heterozygosity for a given generation, is given by the following formula:

$$H_t = H_1(0.5)^{t-1}$$

where H_1 is the level of heterozygosity for generation 1 (equal to 1 in this model), and t denotes the generation. Phenotypic and genotypic response to selection is monitored to beyond generation F_7, where residual heterozygosity is expected to have fallen to less than 1%; $H_7 = (0.5)^6 = 0.0078$.

(2) Outbreeding

Many crop species exhibit mechanisms which aim to promote outcrossing and prevent self-fertilisation. Cross-pollination is an absolute requirement of crops commonly propagated as hybrids, such as maize, and generally a preliminary stage in the breeding of those crops propagated vegetatively, such as sugarcane (*Saccharum* spp.) and potato (*Solanum tuberosum*). Crops maintained as true outbreeding populations, where the naturally outbreeding nature of the crop is utilised by the plant breeder, include rye (*Secale cereale*), sugar beet (*Beta vulgaris*), and alfalfa (*Medicago sativa*).

(2a) Monogamous outbreeding

Monogamy is by far the most commonly utilised method of outbreeding in commercial crop improvement programmes. The female parent plant receives pollen from a single male parent, and all offspring are thus exclusively the products of the two parental genomes. In practice, such parental genomes may be homozygous (for production of hybrids) or heterozygous (particularly at the start of a breeding cycle for clonally propagated species).

In the case of the PSELECT simulation of monogamous outbreeding, as would occur in a controlled experimental outbreeding programme where parents to be mated are first selected, paired, and then strictly isolated to prevent spurious pollination, the selected individuals from each generation are sequentially defined as the female parent. For each female, one of the selected individuals is randomly chosen, in consequence of a system-supplied uniform distribution random deviate, to act as the male parent (Figure 2.3). Self-pollination is excluded. A nested case statement is used to combine the corresponding loci from both parents, and to define the resultant loci in accordance with the rules described for an inbreeding species. All offspring from the female parent are thus the offspring of a single, randomly defined male parent.

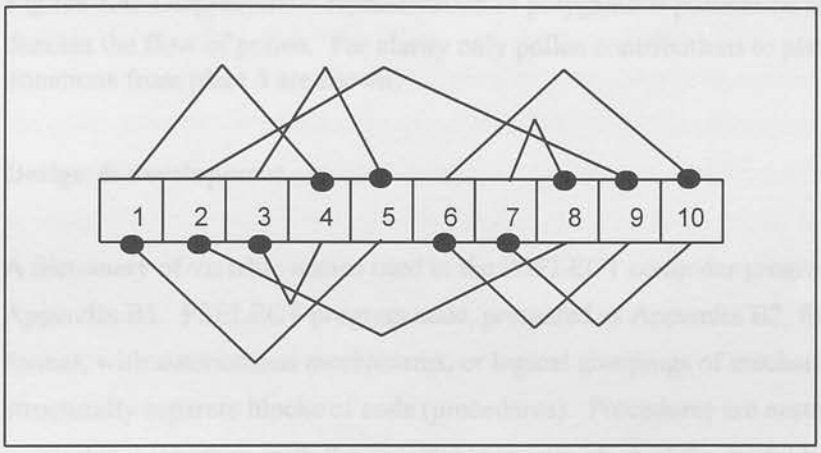


Figure 2.3. Diagrammatic representation of monogamous pollination mechanism. ——● denotes the flow of pollen. For example, plant 1 pollinates plant 4, but is in turn pollinated by plant 5.

(2b) Polygamous outbreeding

Polygamous outbreeding is characteristic of those crops maintained and propagated as outbreeding populations, such as rye and alfalfa. With polygamy, the simulation process is similar to that described for monogamy, except that a designated female will receive pollen from more than one male (Figure 2.4). Each offspring from a sequentially defined female parent is the result of mating with a single randomly specified male parent, hence all alleles within a given offspring come from a single male and single female parent, but each offspring from a given female can be the offspring of different pollen donors. This is achieved by nesting the statement responsible for defining the male parent within that responsible for the regeneration of individual offspring. This would be expected in an experimental situation where selected parents were not isolated at mating, and where no crossing incompatibility is exhibited.

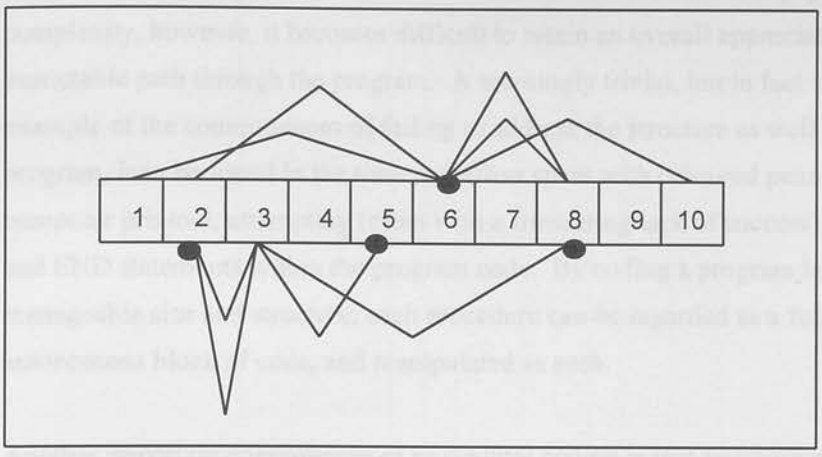


Figure 2.4. Diagrammatic representation of polygamous pollination mechanism. ——● denotes the flow of pollen. For clarity only pollen contributions to plant 6, and pollen donations from plant 3 are shown.

Design & development

A dictionary of variable names used in the PSELECT computer program is listed as Appendix B1. PSELECT program code, presented as Appendix B2, follows a procedural format, with autonomous mechanisms, or logical groupings of mechanisms, coded in structurally separate blocks of code (procedures). Procedures are nested to create a hierarchical structure, with the executable specification of the model being, as far as possible, concentrated into the main program block (physically located at the end of the program). This main program block calls procedures (which may in turn call other procedures) in a sequence which determines executable order, and hence executable flow through the model.

The model did not start as a structured program. As with many of its forerunners , PSELECT is a computer program designed and coded by an individual whose experience was in a field other than computer science, and for whom, at least initially, the application of the model was of primary interest, with structure and design being of little importance beyond the achievement of functionality. In its initial stages, again like some of its forerunners (e.g. BHER and BREED1) the PSELECT program was a single block of code. Limitiations to this approach are obvious, and programmer mental resources become limiting long before system resources. Structured programming is demanding of both time and effort, and would not be expected to yield a satisfactory return on investment in the case of small models, and in this situation the administrative overheads associated with the execution of the procedural program might well result in a performance penalty when

compared to the single-block code. Once code reaches a threshold physical size and complexity, however, it becomes difficult to retain an overall appreciation for the executable path through the program. A seemingly trivial, but in fact very illustrative example of the consequences of failing to address the structure as well as the function of a program, is to be found in the time and effort spent with coloured pens and a pile of Z-fold computer printout, attempting (often with a frustrating lack of success) to balance BEGIN and END statements within the program code. By coding a program into procedures of manageable size and structure, each procedure can be regarded as a functionally autonomous block of code, and manipulated as such.

Another important consequence of procedural coding is that procedures can be called from (administered by) an otherwise empty, and hence unclutered, main program block. Major changes to the functional operation of the model can be effected by very minor changes to the order and content of this main program block. This has obvious benefit for models such as PSELECT, where the program is intended for use as a developmental platform by others. A comprehensive understanding of the workings of the entire model is not required, as at one level, simple changes to the main program block will customise the model to individual requirements, and at another level, provided variable nomenclature is adhered to, straightforward substitution of procedures allows for rival or additional mechanisms to be incorporated into the model.

The central feature of the PSELECT simulation model is the plant population array (Figure 2.5). This takes the form of a 3-dimensional array, with the dimensions X = Number of loci (L), Y = Population size (N), Z = Ploidy (A).

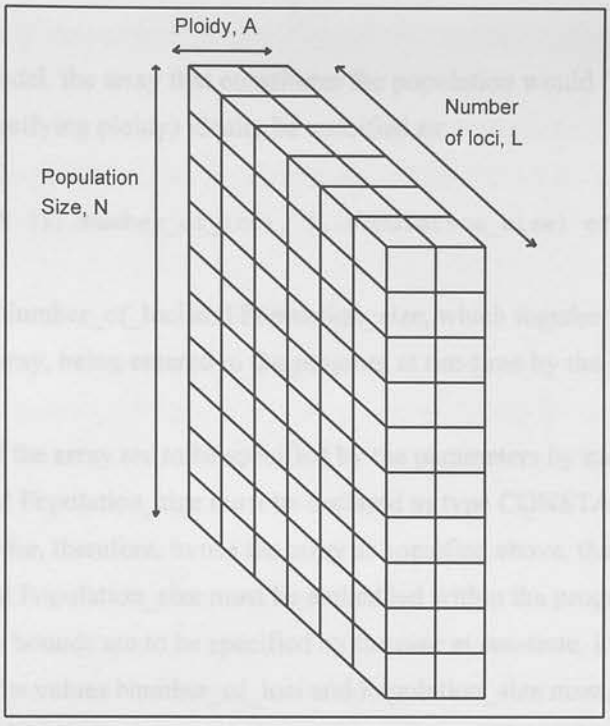


Figure 2.5. Diagrammatic representation of the PSELECT plant population array, illustrating the key parameters ploidy (*A*), number of loci (*L*) and population size (*N*).

The simulated organism is diploid, with up to 20 independently specified loci defining the character under selection (Num_of_loci, $L <= 20$). Population size defines the Y-dimension of the Plant_Population_Array, and maximum population size has been set to 200 ($N <= 200$). All array dimensions have been arbitrarily specified, and there are no technical reasons why larger dimensions cannot be considered. If the values of array indicies are increased, system resources will inevitably become limiting. The point (combination of array index values) at which this occurs has not been established, but could be determined by iterative experimentation.

High level programming languages, almost without exception, permit the declaration of arrays of one or multiple dimensions, where the minimum and maximum index values for each dimension are known at compile-time. There are programming situations, however, where the required size of an array depends upon user-input to the program, with the consequence that array size cannot be determined until run-time. Most high level programming languages do not support run-time specification of array indicies. The standard accepted solution to this situation is to declare an arbitrarily larger-than-required array. This places an upper limit on the quantity of data permitted, and accepts the fact that in most cases a great deal of system memory will be reserved but unused, and hence wasted.

In the PSELECT model, the array that constitutes the population would (ignoring for now the Z-dimension specifying ploidy) ideally be specified as:

```
Population: ARRAY [1..Number_of_loci, 1..Population_size] of INTEGER;
```

with the values for Number_of_loci and Population_size, which together define the X-Y dimensions of the array, being entered to the program at run-time by the user.

If the dimensions of the array are to be specified by the parameters by name, the values for Number_of_loci and Population_size must be declared as type CONSTANT values at compile time. In order, therefore, to use the array as specified above, the values for Number_of_loci and Population_size must be embedded within the program code. In contrast, if the array bounds are to be specified by the user at run-time, ideally in response to screen prompts, the values Number_of_loci and Population_size must, by definition, be declared as type VARIABLE, thus precluding their use in specifying the dimensions of an array.

In Pascal, as in most high-level programming languages, the most straightforward solution to this mutual exclusivity is to specify within the program code, fixed limits to the size of an array. How much of this alotted array size is then used is determined by the user at run-time. The above array would therefore be declared as:

```
Population: ARRAY [1..100,1..1000] of INTEGER;
```

where the maximum permissible value for Number_of_loci is 100, and that for population size is 1000. Clearly, if the user requires a character defined by 5 loci and a population size of 100, the population array represents extremely inefficient use of the finite memory resources of the host system. This has significant consequences for a computer-based model designed for implementation on low budget, generally available personal computer equipment where system resources are likely to be limiting to the operation of the model.

Exceeding system resources is a very real possibility rather than merely an academic consideration. In the developmental phases of the basic model, specification of a value for Num_of_repetitions (i.e. number of program iterations, which constitutes the third dimension of the Results array) in excess of 91 produces the compile-time error message:

Error 96. Too many variables.

which (apart from being illustrative of the generally obscure and unhelpful nature of Turbo Pascal error messages) indicates that the system capabilities have been exceeded, and aborts the compilation of the source code.

A method of interactively specifying array index values is possible if programming in the "C" programming language, through the specification of arrays as abstract data types (Spence, 1993). An abstract data type is one not provided directly by the programming language. By combining standard data types and providing functions to manipulate values of the resultant "abstract" type, implementation details imposed by the compiler can often be circumvented. Whether or not it is advantageous to employ this solution depends on the amount of memory available, how many dynamic arrays are used, and the range of likely array sizes. In any event, what might appear to the user as a frustrating "limitation" might arise from a design feature by the language development team. It should be noted that the deliberate circumvention of a compiler's inbuilt features can have unpredictable and potentially disadvantageous consequences, such as a disproportionately large reduction in speed of execution, unpredictable interaction between data types, or a reduced level of program portability.

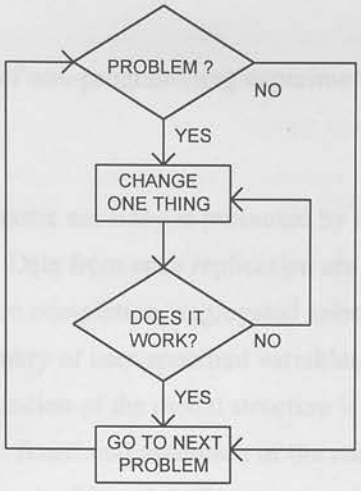
The abstract data type solution is made possible by C's method of memory handling, and is not supported by rival languages. A pointer variable can be used as though it were an array. At run-time it is assumed that the address given by the pointer is that of the first element of the array, and that subsequent elements follow contiguously in memory. The variable is first declared, and a block of memory of the required size is subsequently allocated. The address of the block of memory is then copied to the pointer variable, which can then be used as if it had been declared as an array (equivalent to employing a signpost to say "that's my value over there", rather than bearing a placard saying "my value is this"). In C, a two-dimensional array can be defined by a one-dimensional array of pointers, each pointing to a further one-dimensional array. This means that each element is accessed by following two pointer values, as opposed to using the address arithmetic co-ordinates utilised by most high-level languages. This technique can be extended to arrays of higher dimension, where an n-dimensional array consists of a one-dimensional array of (n-1)-dimensional arrays.

A significant drawback of this technique of array specification is that in the allocation of the block of memory comprising the physical body of the array, the attempt to reserve memory would fail if there is insufficient memory available, leading to the run-time failure of the

entire procedure, which for a nested (hierarchical) procedural structure (such as the mechanistic models considered in this study) would result in complete program failure. An additional, and potentially disastrous disadvantage of the above technique of array specification is that it precludes the checking of array bounds (an intrinsic atribute of address arithmetic methods), and there is therefore no way of verifying that each attempted access to an array element lies within the bounds of the array. Because of the dynamic nature of array specification, it is all but impossible for the compiler to generate such checks (Spence, 1993).

The feasibility of translating the existing Pascal code to C, with a view to obtaining a model where array-limiting parameters can be specified by the user at run-time, was investigated. Public domain software, and "Shareware" products that automatically translate Pascal code to C code, are generally available. A good, but by no means unique example is the P2C utility (Gillespie, 1989). There is no guarantee that such products will generate the most efficient translation of the original code, as in addition to the physical content of the original source code a great deal depends upon its structure. The translated code must be checked, verified, and in the great majority of cases be manually amended before a useable end-product is obtained. This option was not pursued, largely because it was decided that the development of a user interface over and above the level required for model functionality, was primarily a commercial consideration rather than being of academic interest. It is also a widely held and reported opinion that the C programming language has a long learning curve (Rossiter, 1991), and it was felt that further efforts in this direction would have yielded insufficient gain to justify the expenditure of time. Further research in this field might consider the use of C or its object-oriented descendent C++ (Lippman, 1989; Ellis & Stroustrup, 1990) as a modelling tool, although the reader should be warned that, *"C++ is even more cryptic, intimidating, and hard to learn than C"* (Rossiter, 1991).

With regards to the experimental techniques employed in the development of computer-based simulation models, the principal methodology can be effectively summarised by one word, iteration:



An advantage of procedural programming is that, in breaking the program code into small, functionally discrete elements, programming errors can be more effectively located. Upon discovery of a problem within the programming code, and preliminary identification of the procedure within which the fault is arising, the task is to identify exactly the source of the error. The standard way to do this is to insert a statement to write the value of each variable to a debugging file immediately following the calculation of each variable within the procedure. In PSELECT, this file is given the logical name DBug_Out, and the standard form of the test line is as follows:

```
WRITELN(DBug_Out,variable_name);
```

By stepping through each variable in turn, the exact position within the program code at which an error arises can be pinpointed through study of the debugging file. In most case the error will be readily identifiable (typing errors, omitted or inappropriate punctuation etc.), and consequently easily fixed. In other instances the problem will be a semantic or syntactical error arising as a consequence of model structure and resulting in an incorrect sequence of execution. Problems of this nature require careful thought, as the origin of the problem may be physically remote from the point of its manifestation.

As an optional feature of the PSELECT program, a validation file, PSELECT.BUG, giving the full description of each and every population array throughout the course of the simulation, along with a report of key population parameters, can be generated. The file is opened in response to the BOOLEAN variable Debug_On returning a value of TRUE (specified by the user). This option is very expensive in terms of machine resources, and is used only as part of the model verification process.

Specific illustrative examples of non-programming experimentation are presented at the end of this section.

For all breeding regimes, stochastic accuracy is promoted by the ability to specify multiple replications of the simulation. Data from each replication are fed into a three-dimensional storage array, and upon program completion, aggregated selection data (mean values across repetitions), along with a summary of user-specified variables, are output to text file (Figure 2.6). A diagrammatic representation of the model structure is presented (Figure 2.7), as is a schematic representation of the functional execution of the computer program (Figure 2.8). In addition to the aggregated results file, a data file showing the results of each individual simulation run, is produced by the simulation.

Response to selection in a hypothetical plant species

	Population							Selected plants				
F	Genotype		Environment		Phenotype		Het	Genotype		Phenotype		Het
	MEAN	V(G)	MEAN	V(E)	MEAN	V(P)		MEAN	V(G)	MEAN	V(P)	
1	40.0	0.0	-0.0	23.1	40.0	23.1	1.00	40.0	0.0	48.4	3.1	1.00
2	40.0	5.0	-0.0	23.5	40.0	28.3	0.50	41.6	4.0	49.1	3.6	0.49
3	41.6	6.1	-0.1	23.3	41.5	30.4	0.25	43.5	4.1	50.8	4.2	0.23
4	43.6	4.9	0.0	22.8	43.6	27.5	0.12	45.0	3.2	52.6	4.0	0.11
5	45.0	3.4	-0.0	23.0	45.0	26.9	0.05	46.2	2.3	53.8	3.7	0.04
6	46.1	2.2	-0.1	24.0	46.0	26.1	0.02	46.7	1.6	54.9	3.5	0.02
7	46.7	1.5	-0.1	23.2	46.7	24.7	0.01	47.2	0.8	55.1	2.9	0.01
8	47.2	0.8	0.0	22.7	47.2	23.7	0.01	47.4	0.6	55.6	3.3	0.00
9	47.4	0.5	0.1	23.1	47.5	23.7	0.00	47.6	0.4	55.9	3.1	0.00

Population size = 100
Full inbreeding
Additive allelic action at 10 loci
Genotypic background = 20
Selection pressure = 0.90 per generation
Applied environmental standard deviation = 5.00
Environmental generation method = Marsaglia-Bray Polar method
Number of repetitions = 50

	Population							Selected plants				
F	Genotype		Environment		Phenotype		Het	Genotype		Phenotype		Het
	MEAN	V(G)	MEAN	V(E)	MEAN	V(P)		MEAN	V(G)	MEAN	V(P)	
	Standard deviations											
1	0.00	0.00	0.53	2.70	0.53	2.70	0.00	0.00	0.00	0.80	1.56	0.00
2	0.23	0.63	0.51	3.24	0.59	4.19	0.02	0.68	1.79	1.02	1.95	0.05
3	0.71	1.79	0.53	2.63	0.87	4.06	0.03	1.01	2.03	1.10	2.71	0.05
4	1.02	1.78	0.50	2.69	1.05	3.97	0.03	1.26	1.65	1.49	2.71	0.04
5	1.25	1.61	0.46	3.31	1.36	3.82	0.02	1.25	1.64	1.58	2.46	0.03
6	1.26	1.53	0.49	2.80	1.37	3.60	0.01	1.26	1.19	1.62	1.69	0.02
7	1.26	1.11	0.50	3.57	1.30	4.13	0.01	1.27	0.89	1.51	1.70	0.02
8	1.27	0.79	0.41	3.33	1.28	3.62	0.01	1.29	0.59	1.38	1.65	0.01
9	1.29	0.55	0.46	2.72	1.45	2.91	0.00	1.28	0.41	1.56	1.44	0.01
	Standard errors											
1	0.00	0.00	0.07	0.38	0.07	0.38	0.00	0.00	0.00	0.11	0.22	0.00
2	0.03	0.09	0.07	0.46	0.08	0.59	0.00	0.10	0.25	0.14	0.28	0.01
3	0.10	0.25	0.07	0.37	0.12	0.57	0.00	0.14	0.29	0.16	0.38	0.01
4	0.14	0.25	0.07	0.38	0.15	0.56	0.00	0.18	0.23	0.21	0.38	0.01
5	0.18	0.23	0.07	0.47	0.19	0.54	0.00	0.18	0.23	0.22	0.35	0.00
6	0.18	0.22	0.07	0.40	0.19	0.51	0.00	0.18	0.17	0.23	0.24	0.00
7	0.18	0.16	0.07	0.51	0.18	0.58	0.00	0.18	0.13	0.21	0.24	0.00
8	0.18	0.11	0.06	0.47	0.18	0.51	0.00	0.18	0.08	0.19	0.23	0.00
9	0.18	0.08	0.07	0.38	0.21	0.41	0.00	0.18	0.06	0.22	0.20	0.00

Degrees of freedom = 49

Figure 2.6. Example PSELECT output file. Mean response values obtained from 50 repetitions of the simulation, under conditions of self-pollination are presented along with the associated standard errors.

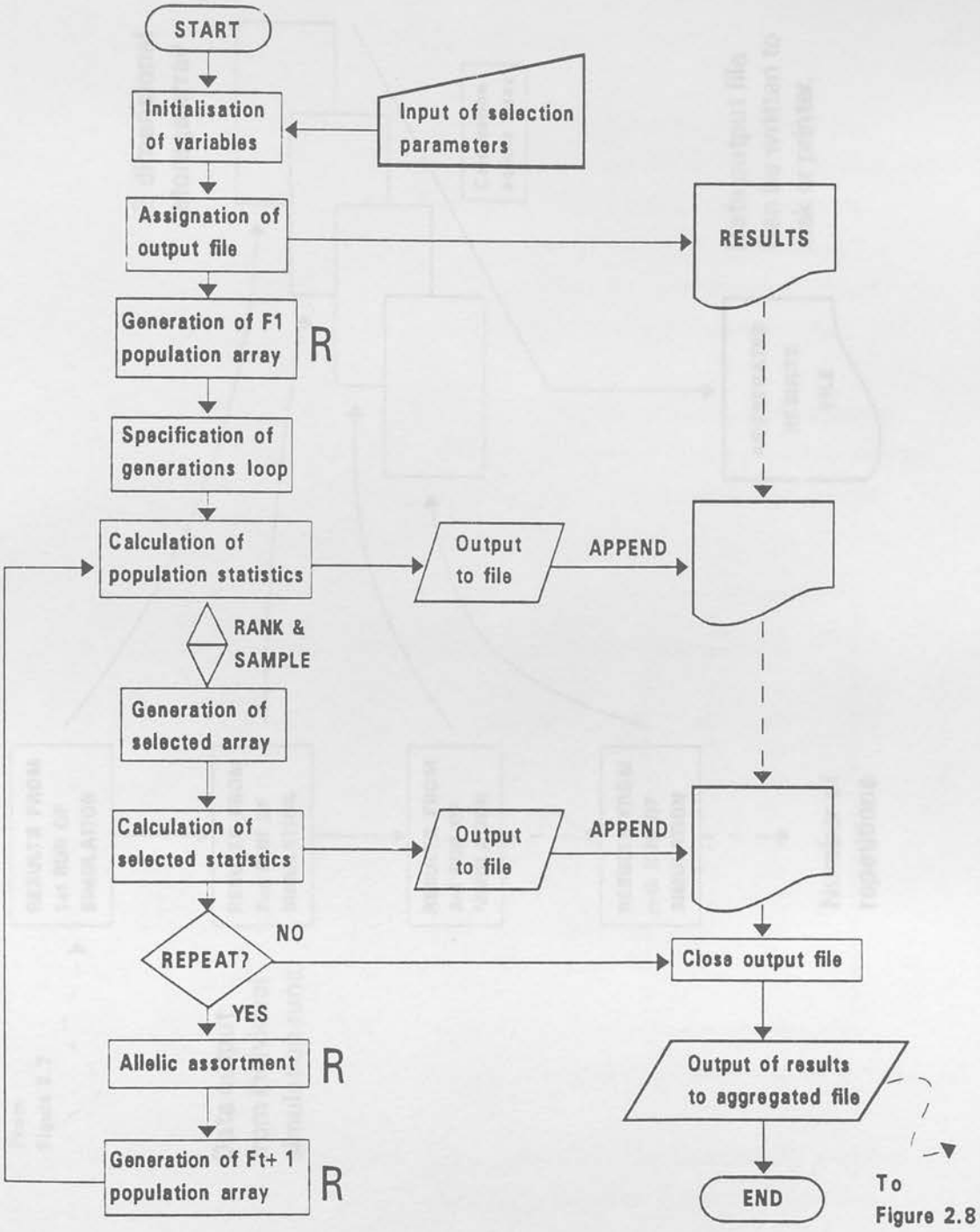


Figure 2.7. Diagrammatic representation of model function, showing the order of program execution, and inputs and outputs to and from the model. R denotes a process in which randomisation is involved.

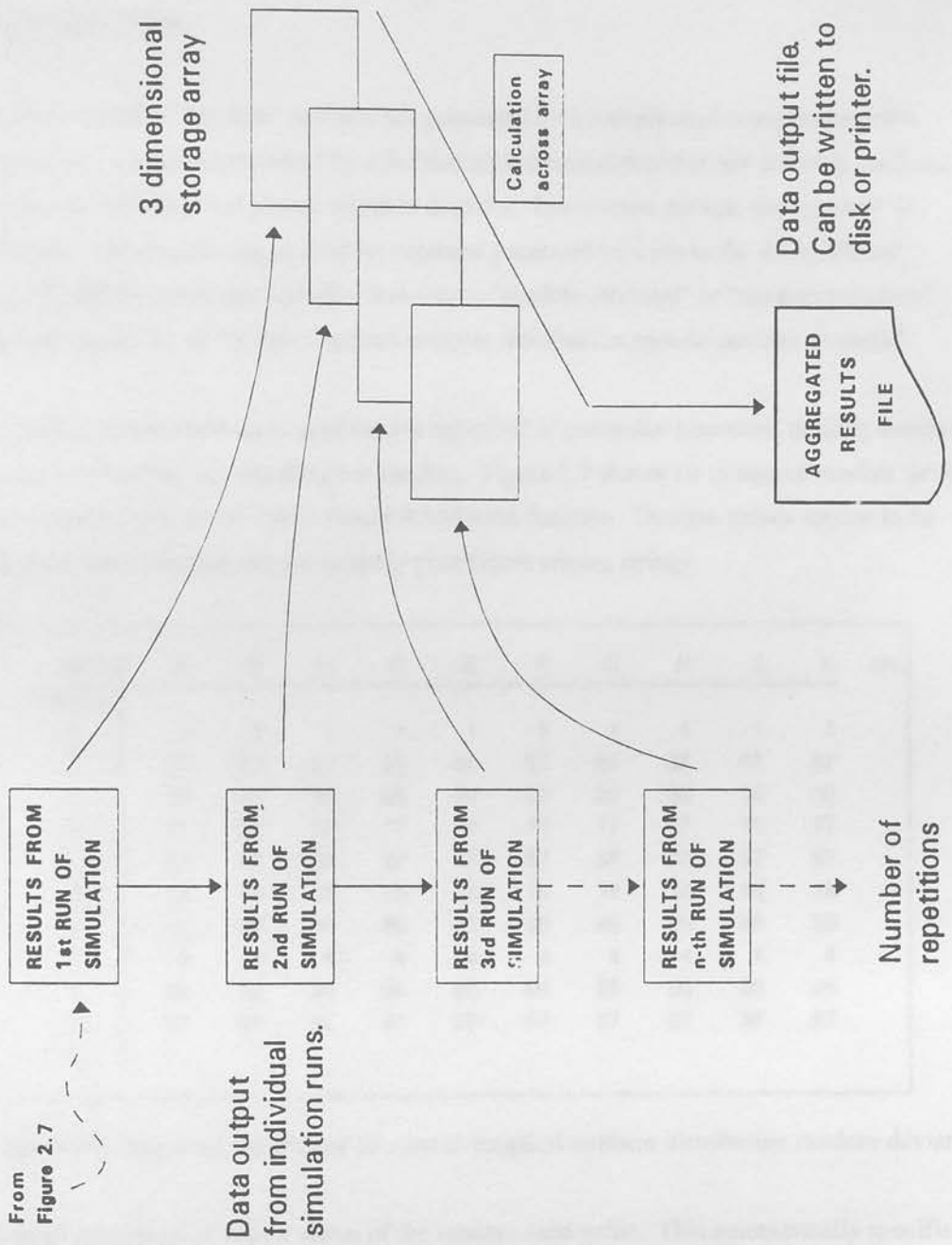


Figure 2.8. Schematic representation of model execution, showing the combination of individual simulation runs to generate aggregated data output.

Example of experimentation 1: verification of system-supplied uniform distribution pseudo-random deviates

INTRODUCTION

System-supplied "random" deviates are generated by a complicated compiler-specific algorithm. Deviates generated by a defined mathematical function are not truly random, and hence are referred to as pseudo-random deviates. In common useage, the "pseudo" is dropped. When referring to random numbers generated by calls to the Turbo Pascal RANDOM function, any and all references to "random deviates" or "random numbers" should strictly be to "system-supplied uniform distribution pseudo-random deviates".

An initial observation upon qualititative appraisal of computer-generated random number strings is that they are anything but random. Figure 2.9 shows 10 strings of random deviates generated by call to the Turbo Pascal RANDOM function. Deviate values appear to be random within strings, but are entirely predictable among strings.

String	A	B	C	D	E	F	G	H	J	K	etc.
Deviate											
1	1	1	1	1	1	1	1	1	1	1	
2	57	57	57	57	57	57	57	57	57	57	
3	30	30	30	30	30	30	30	30	30	30	
4	77	77	77	77	77	77	77	77	77	77	
5	87	87	87	87	87	87	87	87	87	87	
6	18	18	18	18	18	18	18	18	18	18	
7	86	86	86	86	86	86	86	86	86	86	
8	4	4	4	4	4	4	4	4	4	4	
9	96	96	96	96	96	96	96	96	96	96	
10	97	97	97	97	97	97	97	97	97	97	
etc.											

Figure 2.9. Repeated samples of 10 system-supplied uniform distribution random deviates.

This phenomenon arises by virtue of the random seed value. This automatically specified value (system default value = 1) defines the starting point for the RANDOM algorithm. Hence, if all strings share the same starting point, and the same algorithm, it is to be expected that strings would display common deviate values. A randomly specified seed is obtained through call to the Turbo Pascal RANDOMIZE function. Following its specification, different start points within the RANDOM algorithm give unpredictable patterns of deviates across repetitions.

In order to verify that the distribution of system-supplied random deviates within a given system-supplied string is satisfactory, two factors to assess are conformity to the specified mean, and the absence of serial correlation (autocorrelation) between adjacent random deviates.

MATERIALS & METHODS

The computer program RTest01.pas (Figure 2.10) is typical of a simple testing program. After the specification of a RANDOMIZEd seed value, the program calls 1000 random deviates, distributed over the range 1 to 100 inclusive, and writes them to a comma-separated-value format text file, from which they can be imported into an analysis software package, such as Excel (Microsoft Corp.).

```
PROGRAM RTest01;
USES Crt,Dos;
VAR
  RDev100: REAL;
  Data_Output: TEXT;
  Counter: INTEGER;
BEGIN
RANDOMIZE;
ASSIGN(Data_Output,'RTest01.CSV');
REWRITE(Data_Output);
FOR Counter:=1 to 1000 DO
  BEGIN
  RDev100:=(RANDOM(100)+1);
  WRITE(Data_Output,RDev100:6:2);
  WRITELN(Data_Output,',');
  END;
CLOSE(Data_Output);
END.
```

Figure 2.10. Test program RTest01.pas

Summary statistics (Table 2.1) of output from the RTest01 program were used to assess the conformity of system-supplied deviates to the expected mean value (50.5). Standard deviation and standard error values are not presented. Each random number string was duplicated and the copy aligned -1 position out-of-phase with its original, as shown in Figure 2.11. Column 1 by Column 2 correlation was calculated to check for serial correlation between adjacent deviates, R_n against R_{n+1}. Sample size was reduced by 1 due to the loss of the final value in the source string.

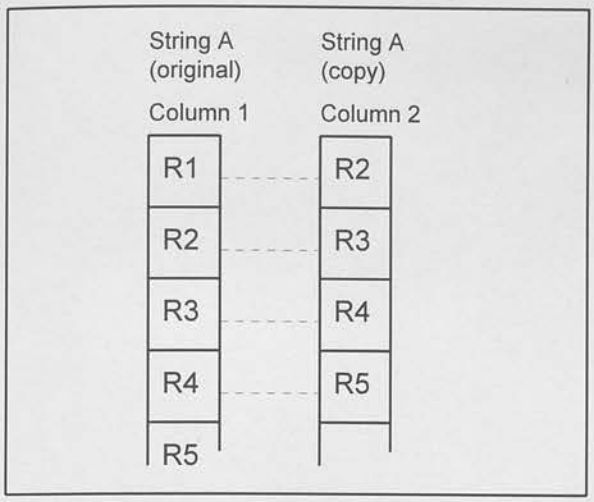


Figure 2.11. Duplication and realignment of computer-generated random deviate strings to check for serial correlation between adjacent deviates in a string.

As a final check of the distribution of random deviates, the sample was ranked, sorted and split into arbitrarily defined intervals (X to X+10 = 10%), and the incidence frequency within each interval assessed and plotted (Figure 2.12) to check for conformity to the expected frequency (= n * 10% = 100).

RESULTS

Three of the 10 sample strings (D, E and H) generated mean values outwith ± 1 standard error (s.e.m.). The largest deviation was seen in sample D (deviation = +1.38 s.e.m.). Serial correlations showed no significance (at 5% level, with n-2 = 997 degrees of freedom) for any sample.

Table 2.1. Replications of 1000 system-supplied uniform distribution random deviates, on the interval 1-100, means, and coefficients of serial correlation between adjacent values.

Replication	A	B	C	D	E	F	G	H	J	K
n	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
Mean	50.76	49.50	50.10	51.75	51.46	51.49	51.39	49.24	51.02	51.00
s.corr.coeff.	0.039	0.014	0.035	0.024	0.025	0.070	0.012	0.019	0.005	0.011

Interval plots for the ten samples showed a qualitative conformity to the expected frequency. The largest deviation from expectation (+28%) was observed for the interval R = 41 to 50 in sample G. Mean incidences across samples were plotted to give a pooled estimate of uniformity of deviate distribution (Figure 2.13).

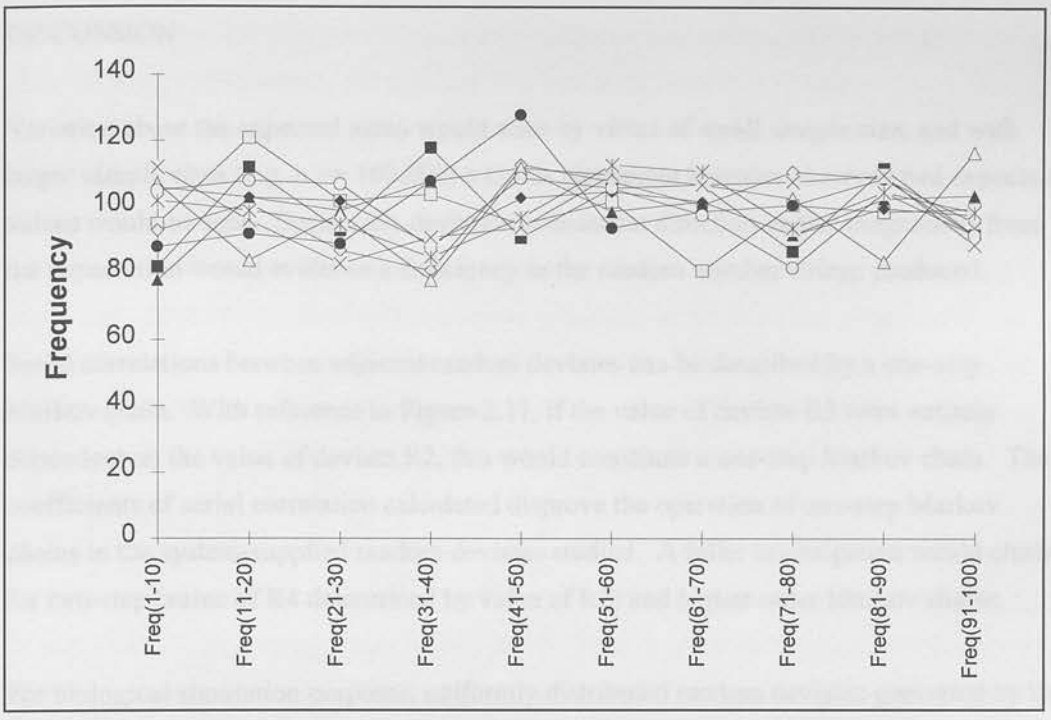


Figure 2.12. Plot of frequency data (intervals of 10) for 10 uniform distribution random deviate strings of sample size 1000.

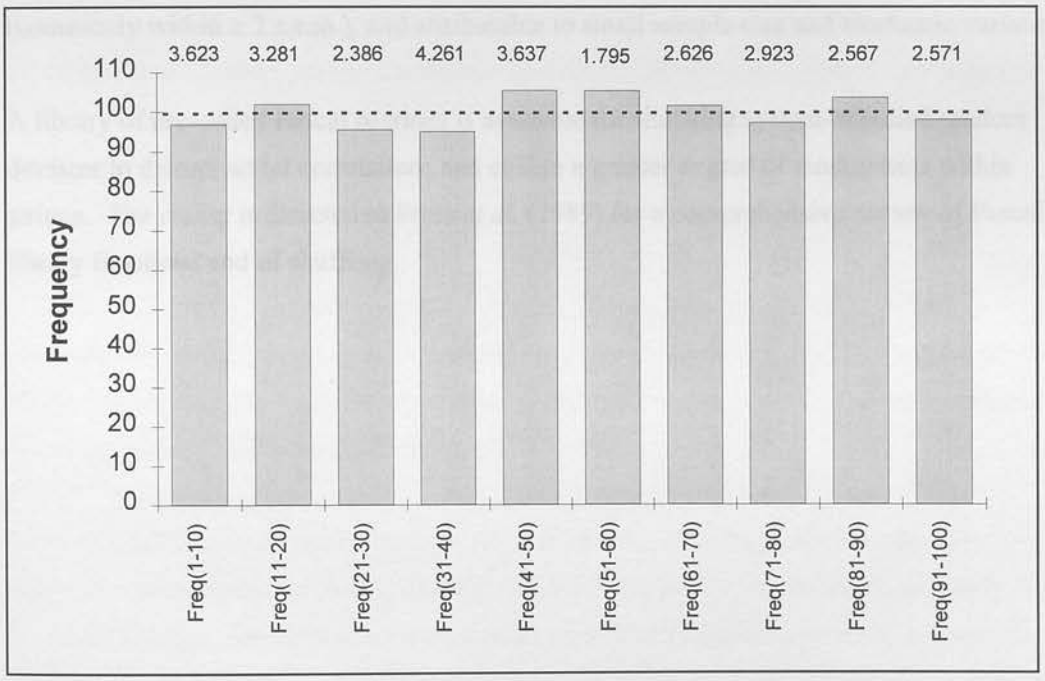


Figure 2.13. Plot of mean frequencies and associated standard errors for interval data, for comparison with expectation of Frequency = 100 for each interval.

DISCUSSION

Variation about the expected mean would arise by virtue of small sample size, and with larger sample sizes (e.g. $n >= 100{,}000$) a closer agreement between observed and expected values would be seen. Systematic deviation (consistent direction and/or magnitude) from the expectation would evidence a deficiency in the random number strings produced.

Serial correlations between adjacent random deviates can be described by a one-step Markov chain. With reference to Figure 2.11, if the value of deviate R3 were entirely dependent on the value of deviate R2, this would constitute a one-step Markov chain. The coefficients of serial correlation calculated disprove the operation of one-step Markov chains in the system-supplied random deviates studied. A fuller investigation would check for two-step (value of R4 determined by value of R2) and higher order Markov chains.

For biological simulation purposes, uniformly distributed random deviates generated by the Turbo Pascal RANDOM function demonstrate satisfactory unpredictability both within and (subject to the use of the RANDOMIZE function) among strings. Deviation from uniform distribution within strings was observed, but deviations were small relative to expectation (commonly within $\pm$ 2 s.e.m.), and attributable to small sample size and stochastic variation.

A library of pre-coded Pascal routines is available for shuffling system-supplied random deviates to disrupt serial correlations and ensure a greater degree of randomness within strings. The reader is directed to Press *et al.* (1989) for a comprehensive review of Pascal library functions and of shuffling.

Example of experimentation 2: comparative assessment of alternative methods for transforming system-supplied random deviates to normal distribution

INTRODUCTION

A key functional stage in the model is the simulation of the environmental contribution to phenotype. It can be assumed that any factor generated by the sum of a large number of independent factors, will be normally distributed. While future developments to the model might have cause to consider alternative distributions, it is reasonable, in this instance, to generate and use a normally distributed simulated environment.

All system-supplied random deviates are uniformly distributed on the specified interval, i.e. if random numbers between 1 and 10 inclusive are specified, there is equal probability that a deviate will be 1 or 6 or any value from 1 to 10. To generate a normally distributed random deviate, system-supplied values have to undergo a transformation. A variety of methods are available for transforming system-supplied uniform distribution random deviates to a normal distribution (Atkinson & Pearce, 1976). Of those that are readily understandable to non-mathematicians, and easily programmed, all so far investigated involve approximation to a lesser or greater degree, which can lead to the distribution generated having significant deviations from normal. In the assessment of rival transformation methods, key points of judgement are conformity of mean (μ) and standard deviation (σ) to specified values, and the absence of serial correlation between adjacent deviates. At a more detailed level of investigation, conformity of the distribution of deviates to the specified normal distribution must be established.

Three transformation methods were considered: Box-Muller transformation (Box & Muller, 1958), Marsaglia-Bray Polar transformation (Marsaglia & Bray, 1964), and transformation according to the Central Limit Theorem (Taha, 1976).

The Box-Muller transformation method (Box & Muller, 1958) centres upon the pairwise generation of independent pseudo-random variables, U_1 and U_2, distributed uniformly on the interval 0 to 1, which together define polar co-ordinates uniformly distributed on the interval 0 to 2π. Inversion of the distributions returns two normally distributed random deviates, X_1 and X_2, having unit variance and a mean distribution of zero. Thus:

$X_1 = (-2\log_e U_1)^{1/2} \sin(2\pi U_2)$

$X_2 = (-2\log_e U_1)^{1/2} \cos(2\pi U_2)$

In Marsaglia-Bray Polar transformation, the normally distributed random deviate is generated as a result of a uniform probability function. The transformation method is complicated, but can be summarised by stating that 97% of the output is generated by arithmetic summation of a string of independent uniform distribution random variables (a mechanism used in Central Limits transformation). The remaining 3% of normal deviates are generated in a pairwise manner, where the normally distributed deviates are given as follows:

$X_1 = U_1[-2(\log_e \beta)/\beta]^{1/2}$

$X_2 = U_2[-2(\log_e \beta)/\beta]^{1/2}$

where U_1 and U_2 are uniformly distributed on the interval -1 to +1, and $\beta = (U_1^2 + U_2^2)$ conditioned by $\beta < 1$.

The Central Limit Theorem dictates that the sum of a large number of independent random variables will define a point on a normal distribution curve, regardless of the original distribution of the random variables. The normally distributed random deviate, X, is given as follows:

$X = [\sigma/(n/12)^{1/2}]\tau - (n/2)$

where σ is the required standard deviation of the distribution, and τ is the sum of the n random deviates generated. τ is asymptotically normal with mean = $n/2$ and variance = $n/12$. n is by convention set at 12 (Taha, 1976).

MATERIALS & METHODS

The computer program RTest04.pas (Appendix B3) generates normally transformed random deviates. The program was used to test for serial correlation between adjacent deviates by generating 100 deviates distributed about a mean of zero with standard deviation of 5, using each of the rival transformation methods. The output data for each transformation were aligned one position out-of-phase (e.g. Figure 2.11) and assessed for serial correlation. X,Y scatter plots of program output, giving a qualitative appraisal of serial correlations, are presented (Figure 2.14).

Program RTest04 was amended to generate samples of 1000 for each transformation method. Data was partitioned into single whole number intervals (centred about the integer, ± 0.5) to present a direct graphical comparison of the output from the three transformation methods (Figures 2.15a-2.15c), and to assess conformity of output data to the specified normal distribution. Mean and standard deviation of the observed distributions were assessed directly from the data. Conformity to the normal distribution was calculated by the Minitab (Release 9 for Windows) NSCORES function (Minitab Inc.).

RESULTS

Output from the three transformation methods were qualitatively comparable. Scatter plots (Figure 2.14) of correlation data from all transformation methods clustered about the origin and showed no discernable internal pattern.

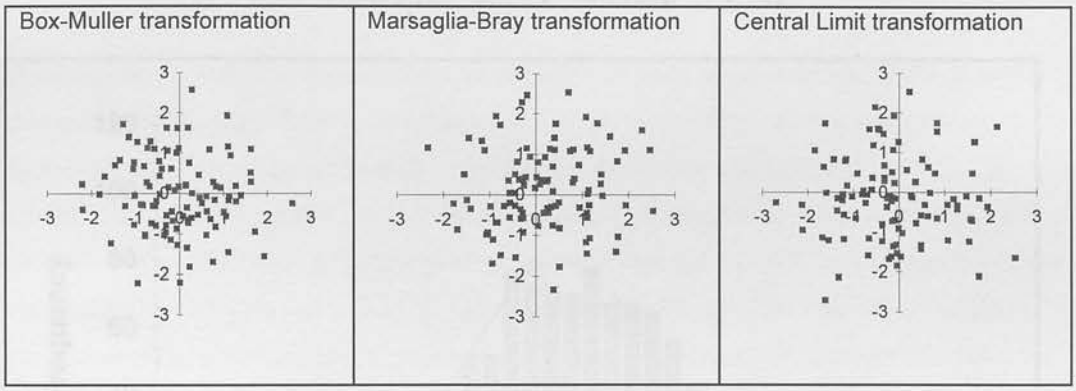


Figure 2.14. Plots of adjacent random deviates, R_n (X) against R_{n+1} (Y), for alternative uniform to normal distribution methods, as an indication of serial correlation between adjacent deviates.

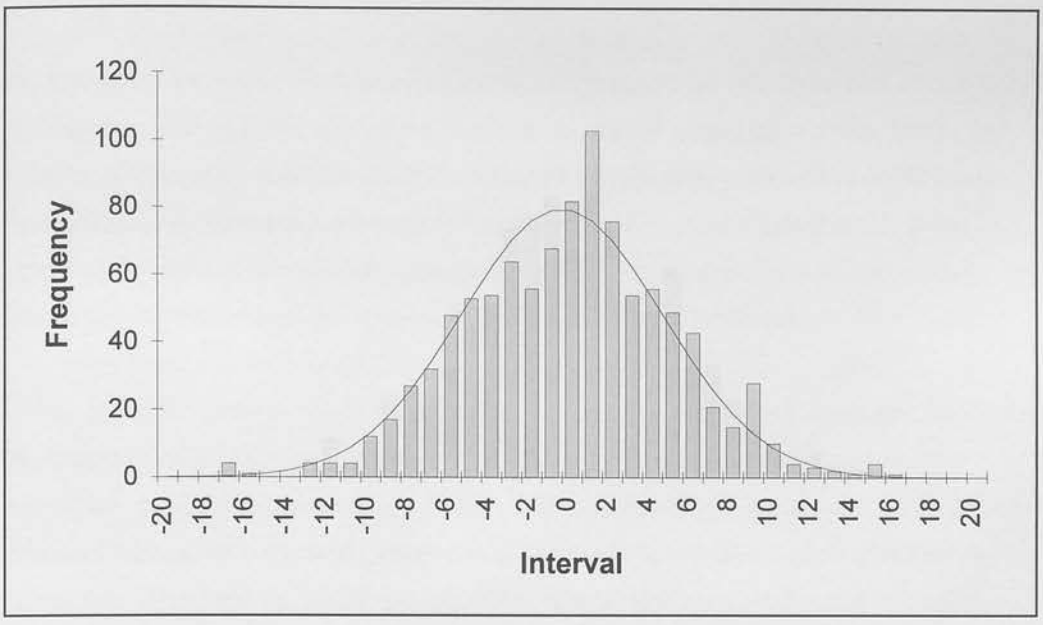


Figure 2.15a. Box-Muller transformation of system-supplied uniform distribution deviates (n = 1000 ~N(0,5), μ = -0.09, σ = 4.972, p >= 0.95).

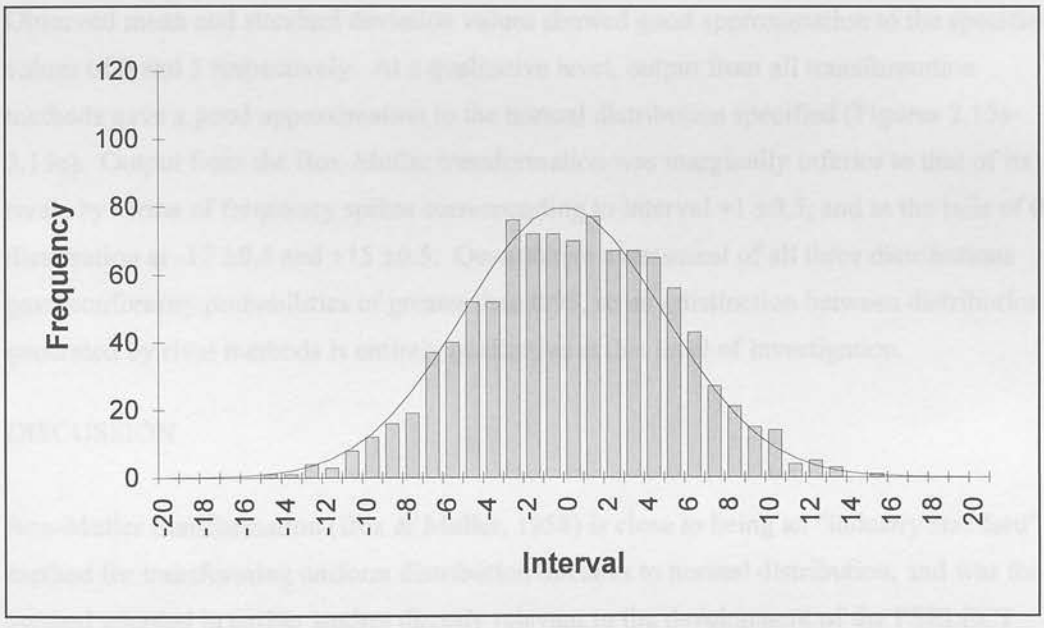


Figure 2.15b. Marsaglia-Bray Polar transformation of system-supplied uniform distribution deviates (n = 1000 ~N(0,5), μ = 0.02, σ = 4.955, p >= 0.95).

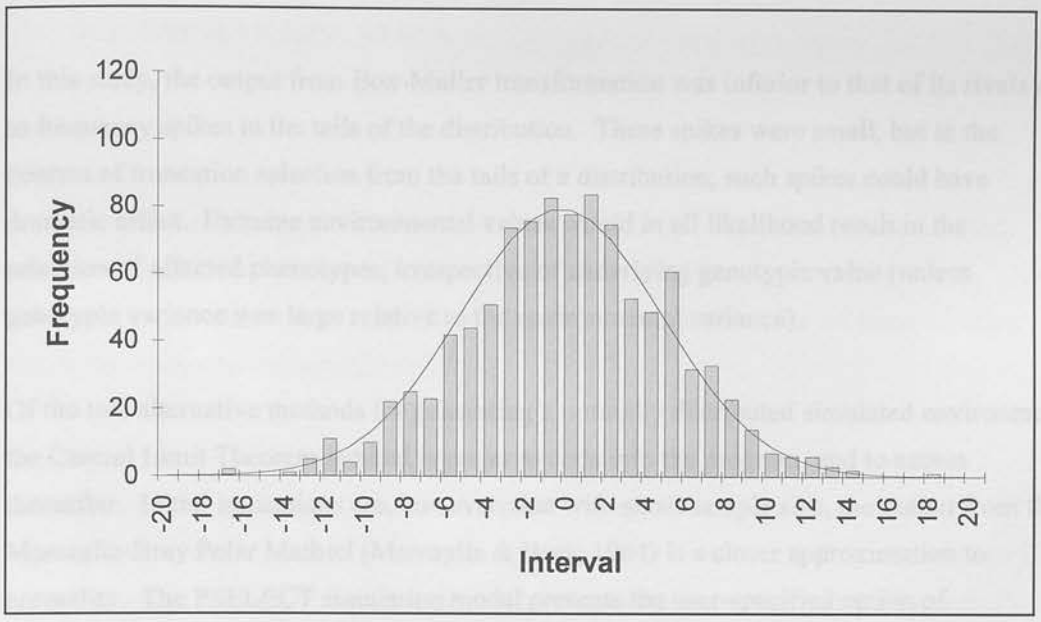


Figure 2.15c. Central Limit Theorem transformation of system-supplied uniform distribution deviates (n = 1000 ~N(0,5), μ = -0.12, σ = 5.045, p >= 0.95).

Observed mean and standard deviation values showed good approximation to the specified values of 0 and 5 respectively. At a qualitative level, output from all transformation methods gave a good approximation to the normal distribution specified (Figures 2.15a-2.15c). Output from the Box-Muller transformation was marginally inferior to that of its rivals by virtue of frequency spikes corresponding to interval +1 ±0.5, and in the tails of the distribution at -17 ±0.5 and +15 ±0.5. Quantitative assessment of all three distributions gave conformity probabilities of greater than 0.95, so any distinction between distributions generated by rival methods is entirely qualitative at this level of investigation.

DISCUSSION

Box-Muller transformation (Box & Muller, 1958) is close to being an "industry standard" method for transforming uniform distribution deviates to normal distribution, and was the method adopted in earlier studies directly relevant to the development of the PSELECT model, such as the BREED1 simulation model (Open University, 1987). Detractors of the Box-Muller method have noted that rather than defining a circle, output from the transformation defines a spiral (Bratley *et al.*, 1987; Law & Kelton, 1991), and in consequence of the progressively increasing potential range of X_1 and X_2, approximation to normality is poor. Spikes in the tails of the generated normal distribution have been noted (Neave, 1973). This has significant effect when sampling from the tails of a distribution, particularly when the simulated data set is large, or the selection pressure is high.

In this study, the output from Box-Muller transformation was inferior to that of its rivals due to frequency spikes in the tails of the distribution. These spikes were small, but in the context of truncation selection from the tails of a distribution, such spikes could have dramatic effect. Extreme environmental values would in all likelihood result in the selection of affected phenotypes, irrespective of underlying genotypic value (unless genotypic variance was large relative to the environmental variance).

Of the two alternative methods for generating a normally distributed simulated environment, the Central Limit Theorem method is easier to code into the program, and to access thereafter. Initial indications are, however, that with small sample size, the output from the Marsaglia-Bray Polar Method (Marsaglia & Bray, 1964) is a closer approximation to normality. The PSELECT simulation model presents the user-specified option of generating environmental variance by the Box-Muller method, the Central Limit Theorem or the Marsaglia-Bray Polar method. The method used will depend upon the objectives of, and the assumptions made by the model user.

VALIDATION & VERIFICATION

Model output

In its present form, and upon the PS/2-55sx computer system described, the maximum number of repetitions that can successfully compile was iteratively determined as 91. The program compiles in seconds, and full execution of 50 repetitions, for all three breeding protocols, takes approximately 2 minutes.

Results are presented for selection for increased phenotypic score in the cases of monogamous outbreeding, polygamous outbreeding and inbreeding, under differing environmental conditions. Population size (N) was kept constant at 100, the number of loci (L) defining the character under selection was 10, with a constant background genotypic value of 20 units. Unless stated, selection pressure (P_s) was 0.9 per generation, and applied environmental standard deviation (esd) was 5 about a mean of 0. Arithmetic means of results obtained from 20 repetitions of the model for each parameter assessment were used to generate graphical representations of phenotypic and genotypic responses to selection. For the sake of clarity, standard error values are not presented.

Genotypic vs. Phenotypic response to selection

Responses of mean genotype and mean phenotype to selection (Figure 2.16) showed a high degree of correlation ($\gamma_{(G\text{-}P)} >= 0.998$) which is in line with expectation. Respective variances were not comparable. In light of the high correlation between responses, only genotypic response was subsequently considered.

Allelic dominance relationships

Response patterns followed the expectation (Figure 2.17). F_1 heterosis was observed where dominance was operating, with partial dominance resulting in an intermediate level of F_1 heterosis. Selection responses were initiated at generation 2, corresponding to the introduction of genotypic variance to the population. Thereafter, selection responses were comparable.

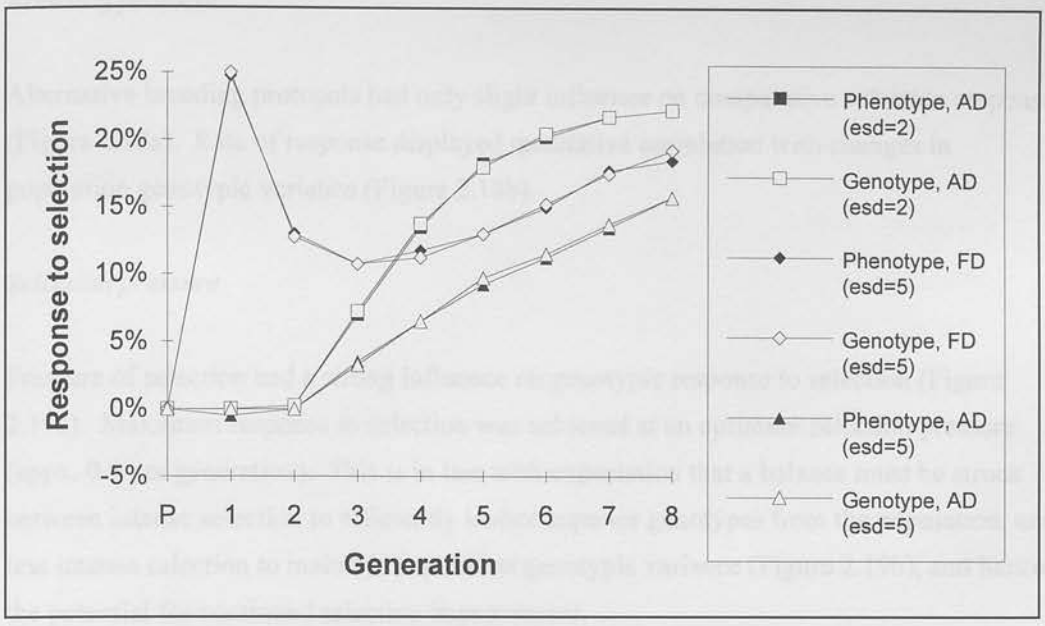


Figure 2.16. Comparison of phenotypic and genotypic responses to selection, under different allelic dominance regimens (AD = additive dominance, FD = full dominance) and different levels of esd. $N = 100$, $L = 10$, propagation by self pollination, $P_s = 0.9$ per generation.

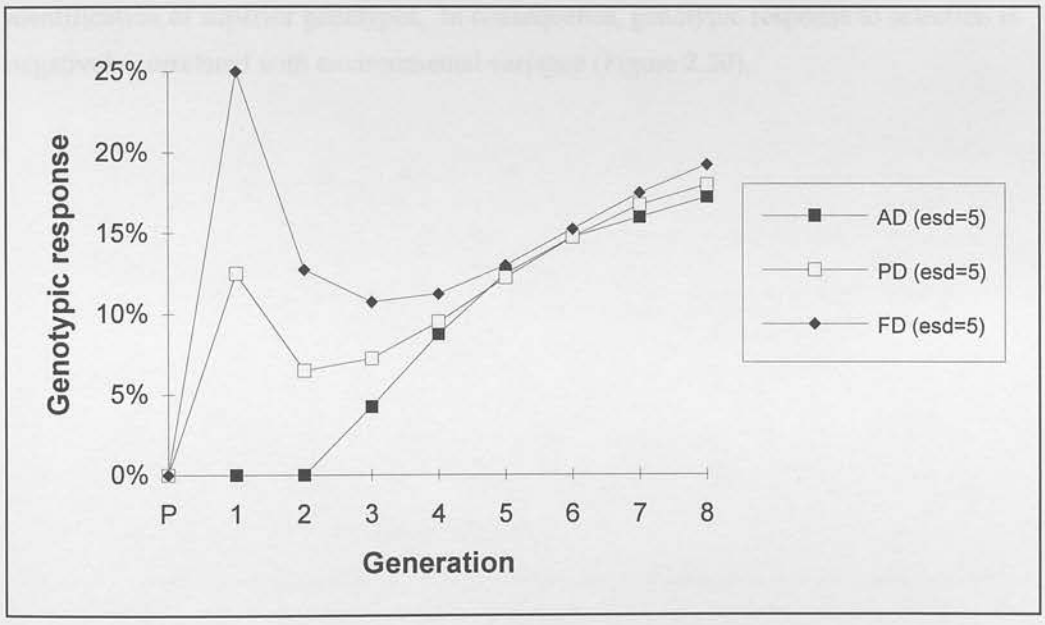


Figure 2.17. Comparison of genotypic response to selection for different allelic dominance relationships (AD = additive dominance, PD = partial dominance, FD = full dominance). Genotypic value of $A_1A_2 = 0.75(A_1A_1)$. $N = 100$, $L = 10$, esd = 5 about a mean of 0, propagation by monogamous outbreeding, $P_s = 0.9$ per generation.

Breeding protocol

Alternative breeding protocols had only slight influence on comparative selection responses (Figure 2.18a). Rate of response displayed qualitative correlation with changes in population genotypic variance (Figure 2.18b).

Selection pressure

Pressure of selection had a strong influence on genotypic response to selection (Figure 2.19a). Maximum response to selection was achieved at an optimum selection pressure (appx. 0.9 per generation). This is in line with expectation that a balance must be struck between intense selection to efficiently isolate superior genotypes from the population, and less intense selection to maintain population genotypic variance (Figure 2.19b), and hence the potential for continued selective improvement.

Environmental variation

Environmental variance contributes to phenotypic variance and hence inhibits the efficient identification of superior genotypes. In consequence, genotypic response to selection is negatively correlated with environmental variance (Figure 2.20).

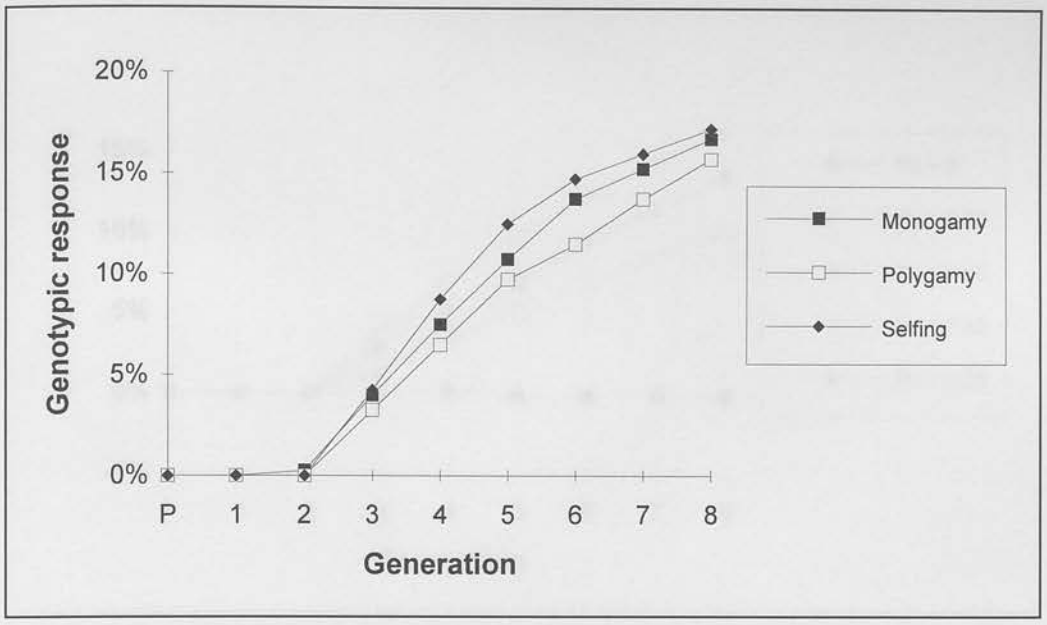


Figure 2.18a. Comparison of genotypic response to selection for different breeding protocols. $N = 100$, $L = 10$, additive dominance at all loci, esd = 5 about a mean of 0, P_s = 0.9 per generation.

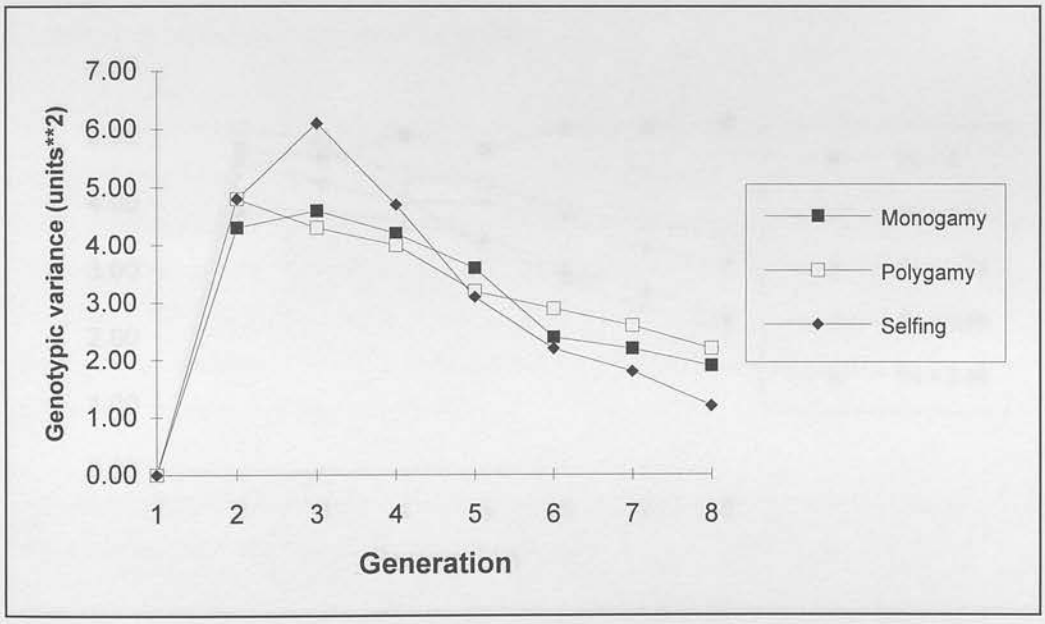


Figure 2.18b. The maintenance of population genotypic variance (V_G) under different breeding protocols. $N = 100$, $L = 10$, additive dominance at all loci, esd = 5 about a mean of 0, $P_s = 0.9$ per generation.

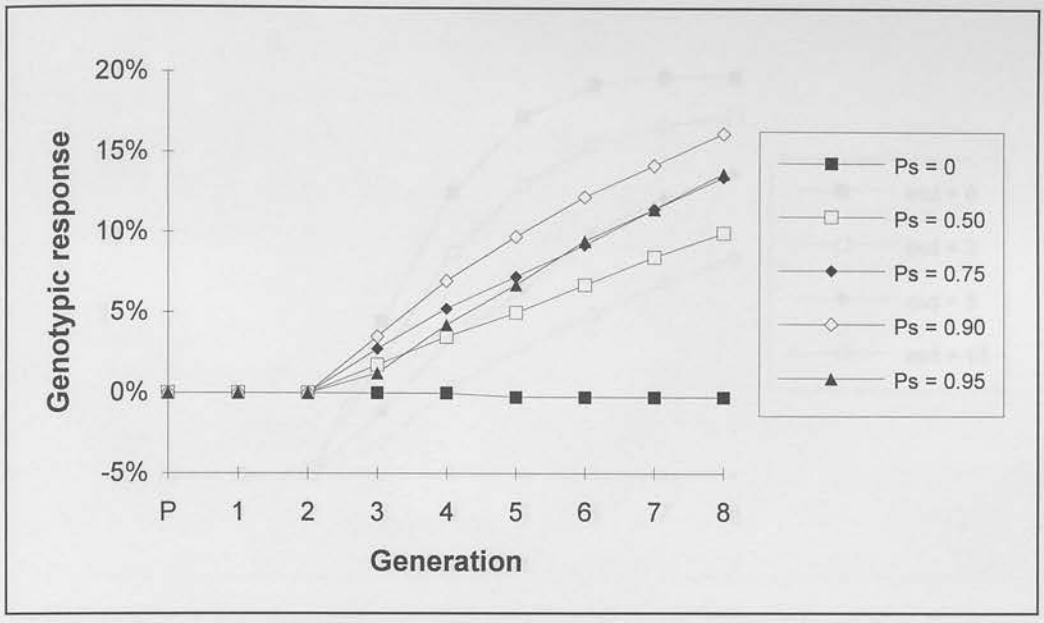


Figure 2.19a. Comparison of genotypic response to selection for different selection pressures. $N = 100$, $L = 10$, additive dominance at all loci, esd = 5 about a mean of 0, propagation by polygamous outcrossing.

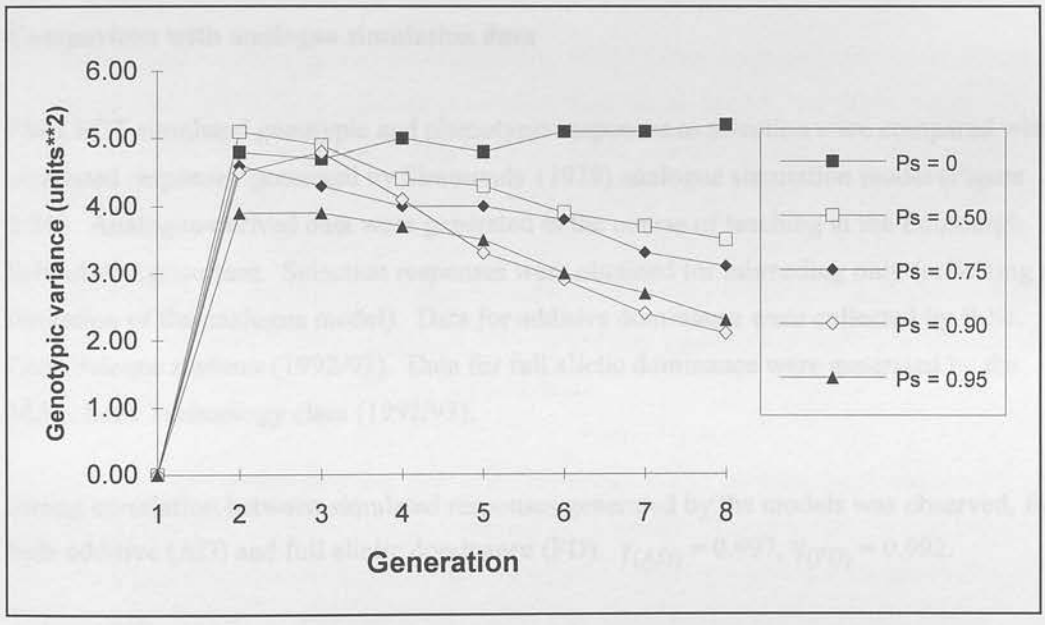


Figure 2.19b. The maintenance of population genotypic variance (V_G) under different selection pressures. $N = 100$, $L = 10$, additive dominance at all loci, esd = 5 about a mean of 0, propagation by polygamous outcrossing.

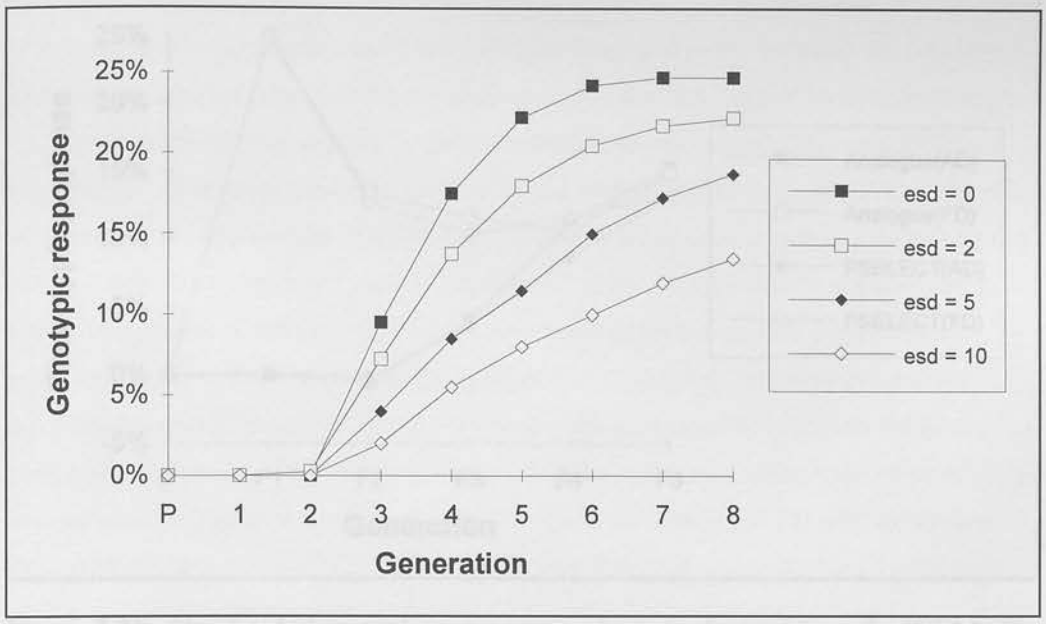


Figure 2.20. Comparison of genotypic response to selection for different levels of applied environmental standard deviation (esd). $N = 100$, $L = 10$, additive dominance at all loci, propagation by self-pollination, $P_s = 0.9$ per generation.

Comparison with analogue simulation data

PSELECT simulated genotypic and phenotypic responses to selection were compared with simulated responses generated by Simmonds' (1979) analogue simulation model (Figure 2.21). Analogue-derived data were generated in the course of teaching at the Edinburgh School of Agriculture. Selection responses were obtained for inbreeding only (reflecting a limitation of the analogue model). Data for additive dominance were collected by B.Sc. Crop Science students (1992/93). Data for full allelic dominance were generated by the M.Sc. Seed Technology class (1992/93).

Strong correlation between simulated responses generated by the models was observed, for both additive (AD) and full allelic dominance (FD). $\gamma_{(AD)} = 0.997$, $\gamma_{(FD)} = 0.992$.

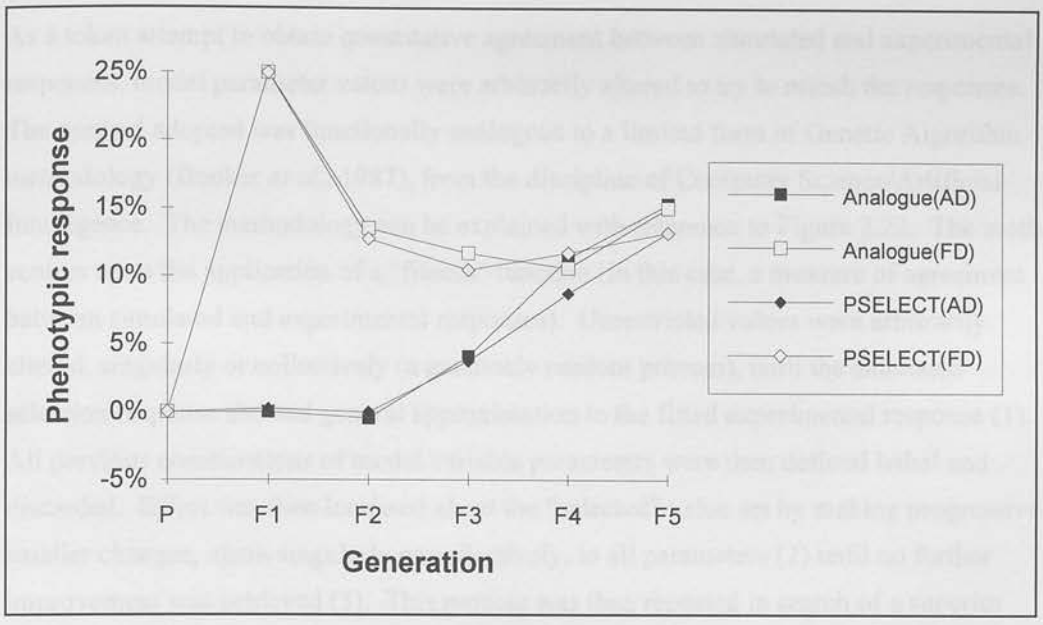


Figure 2.21. Simulated phenotypic responses to selection, obtained from the PSELECT model and Simmonds' analogue simulation model.

Comparison with experimental data

Validating model output against experimentally-derived data presents a dilemma. When validating a set of results, a standard is required with which to compare those results. In comparing PSELECT output with data from an experimental breeding programme selecting for reduced plant height in Brassicas (described in thesis section III), it is not possible to state which data set constitutes the standard against which the other should be validated. Quantitative inferences from the comparison are meaningless without an authenticated point of reference, and in the absence of such the best that can be achieved is the establishment of broad qualitative agreement between the data sets.

PSELECT parameters population size, selection pressure and breeding protocol can be matched to those of the experimental breeding programme. However, the genetic basis of plant height in *Brassica carinata* and *Brassica juncea* (the experimental material) is not known. The PSELECT parameters number of loci, genomic background, dominance relationships, number of alleles segregating within the population and their relative contributions to genotype, can only, therefore, be arbitrarily set. A true sensitivity analysis, to determine the role of each parameter, is precluded by virtue of multiple unknown parameters.

As a token attempt to obtain quantitative agreement between simulated and experimental responses, model parameter values were arbitrarily altered to try to match the responses. The method adopted was functionally analogous to a limited form of Genetic Algorithm methodology (Booker *et al.*, 1987), from the discipline of Computer Science/Artificial Intelligence. The methodology can be explained with reference to Figure 2.22. The method centres upon the application of a "fitness" function (in this case, a measure of agreement between simulated and experimental responses). Unrestricted values were arbitrarily altered, singularly or collectively (a genuinely random process), until the simulated selection response showed general approximation to the fitted experimental response (1). All previous combinations of model variable parameters were then defined lethal and discarded. Effort was then localised about the "selected" value set by making progressively smaller changes, again singularly or collectively, to all parameters (2) until no further improvement was achieved (3). This process was then repeated in search of a superior (fitter) response correlation (4, 5, 6 etc.). The method described differs from true Genetic Algorithms in that mechanisms functionally analogous to mutation and crossing-over are not included.

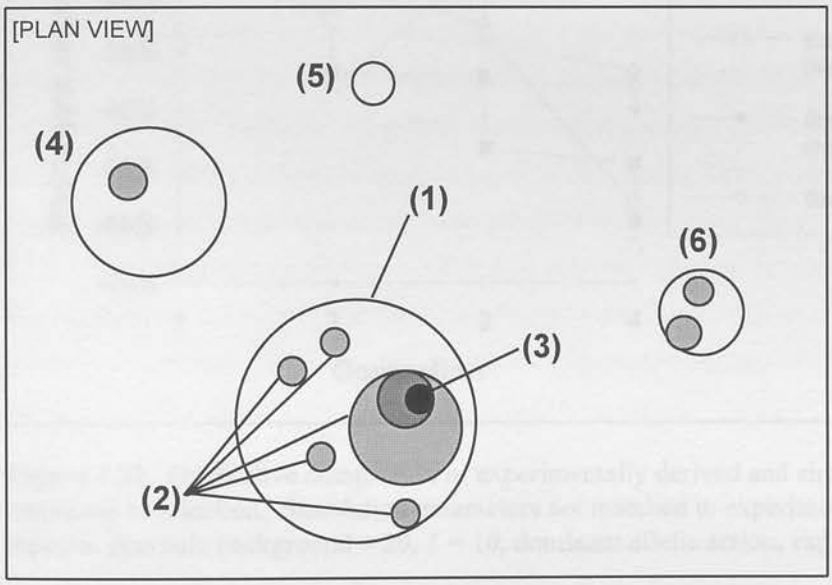


Figure 2.22. Abstract representation of adapted Genetic Algorithm methodology, with density of colour representing proximation of simulated response to fitted experimental response.

The resulting situation is similar in many ways to the concept of adaptive (selective) landscapes (Wright, 1932), and the attainment of "false peaks". Improved correlation between responses will be achieved up to a point, which in all but one case will fall short of the potential maximum. The only solution to this is continued trial and error.

The best quantitative agreement ($\gamma_{(selfing)} = 0.987$, $\gamma_{(polygamy)} = 0.993$) achieved prior to the abandonment of this study was with the following parameter values: $N = 20$, $P_s = 0.8$ (both fixed by experimental protocol; $L = 5$, full allelic dominance at all loci, esd = 0.75 about a mean of 0. Correlations are good, but as this is a contrived result no inferences can be drawn regarding the genetic basis of plant height in Brassicas.

In the absence of a mechanism for quantitative validation, qualitative agreement between patterns of response was sought. Phenotypic mean response (including mean values for the selected proportion of each generation) was plotted against experimental data (Figure 2.23). Standard error values for experimental data ranged from 0 (F_1 selected) to 5.23% (F_2 polygamy).

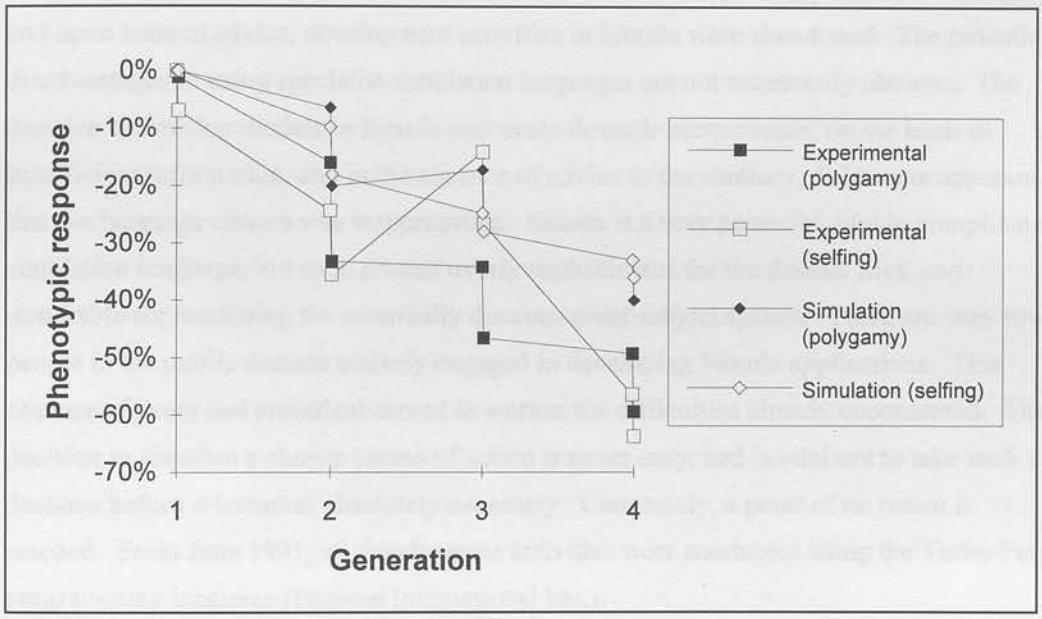


Figure 2.23. Qualitative comparison of experimentally derived and simulated phenotypic responses to selection. Simulation parameters not matched to experimental protocol were as follows: genomic background = 20, $L = 10$, dominant allelic action, esd = 0.5, single runs.

Experimental and simulated phenotypic responses showed good qualitative agreement: $\gamma_{(polygamy)} = 0.989$, $\gamma_{(selfing)} = 0.835$. This must, however, be considered in light of demonstrated high correlation within method, irrespective of breeding protocol: $\gamma_{(experimental)} = 0.864$, $\gamma_{(simulation)} = 0.973$.

DISCUSSION

Response data generated by PSELECT demonstrates good qualitative agreement with expectation. In addition, response data generated by the PSELECT model showed a high quantitative correlation with data from its analogue model predecessor. Observed qualitative agreement between PSELECT response data and experimentally-obtained responses was satisfactory, but in the absence of independent verification of either data set, no quantitative conclusions can be made.

PSELECT was originally developed in the object-oriented computer simulation language PC-Simula (Simula a.s.). Progress was slow, reflecting the perceived difficulty in the use of this language by non-computer professionals (Bratley *et al.*, 1987). A prototype version of PSELECT was presented (Partner *et al.*, 1991), but after disappointing subsequent progress, and upon learned advice, development activities in Simula were abandoned. The potential disadvantages in using specialist simulation languages are not necessarily obvious. The decision to develop models in Simula was made through inexperience, on the basis of insufficient information, and in the absence of advice to the contrary. It become apparent that the language chosen was inappropriate. Simula is a very powerful, highly complicated simulation language, but soon proved overly sophisticated for the desired uses, and unsuitable for modelling the essentially discrete-event subject system. There are very few people in the public domain actively engaged in developing Simula applications. This absence of peers and precedent served to worsen the difficulties already encountered. The decision to abandon a chosen course of action is never easy, and is vital not to take such a decision before it becomes absolutely necessary. Conversely, a point of no return is reached. From June 1991, all development activities were conducted using the Turbo Pascal programming language (Borland International Inc.).

As a developmental academic model, PSELECT currently does not feature a sophisticated user interface to aid the specification of selection parameters. If a model is to be used it must be of use, and if a model is to be of use it must be usable. Effort was not directed to the development of a user-interface for PSELECT as this feature would not be of primary academic interest. This notwithstanding, if any model is to be adopted for use, the interface between the model and the model user is crucial. The current user-model interface guides the user through the model data input routines, but offers little flexibility with respect to altering or reviewing values once entered. A seemingly trivial point, but one which has a disproportionate influence on the adoption or rejection of a model for general use is that the

current PSELECT interface is aesthetically uninspiring. An uninterested potential user is unlikely to become an enthusiastic user.

Future developments to the model will need to address the user-interface as well as scientific aspects of model structure and function. Utilities allowing non-computer professionals to design and develop computer system interfaces are generally available, and preliminary investigation has established two packages, The Laughing Dog ScreenMaker (Yardbird Software) and the Technojock Object Toolkit (Technojock Software Inc.), that work well with Turbo Pascal coded programs. The Technojock Object Toolkit found implementation in the recently presented apple scab simulation model VENTEM (Butt & Xu, 1993).

With respect to the model structure, the majority of input parameters are defined variables, with values being specified by the user at model run-time. The values presented for the relative genotypic contributions of alleles, the number of loci defining the selected character, environmental variance, population size and the proportion of individuals selected, were chosen with the objective of simplification of the model validation procedures. In its current form, the model does not claim to represent any specific experimental situation, but was designed as a theoretical mechanistic model, for use as a developmental platform in the design and construction of more sophisticated models.

In this study, 50, 20 and 10 repetitions were chosen as arbitrary figures for validation procedures. Although the issue has not yet been addressed experimentally, initial impressions are that the results obtained from both 50 and 91 repetitions of the simulation, are not significantly different from those obtained from a lower level of repetition, e.g. 10 repetitions. This is an area where continued investigation might prove profitable, particularly with the expectation that increasing model sophistication and complexity will begin to test the capabilities of the host computer system.

On a biological level, the basic model could be improved in various ways. Facility should be made for some loci exhibiting additive dominance, while others would exhibit varying degrees of classical allelic dominance. The final outcome of this would be a model where degree of dominance could be specified for each locus. Clearly not all loci will be of equal effect and there is a need to be able to vary the relative contributions to the genotype conferred by each locus. It cannot be assumed that all loci will be mutually independent in their action, and later developments will need to reflect this. In population terms, it is unlikely that only two allelic forms will be present. Clearly, the model would be improved

by the ability to simulate more than two alleles in the population. At a higher level of organisation, the model would need to be extended to simulate simultaneous selection for more than one character.

Evolutionary forces rarely act in isolation, and with the exception of certain experimental situations, more than one agent of genetic change will be acting upon a population at any given time. Established predictive models in population genetics generally use observations from the system under consideration, to derive an empirical model, typically equation-based, to estimate the net changes in the system. Conversely, if each evolutionary force is treated in isolation, and then used as a component of the net change in gene frequencies, it is possible to construct a mechanistic model that reflects the effect of individual genetic processes, a synthesis of which will then give an estimate of net change attributable to all forces. Dispersive processes will operate in the model population, in consequence of finite population size and restricted sample size. Holding population size and selection pressure constant among generations allows the assumption that the magnitude of dispersive processes will be constant. No assumptions regarding the direction of this deviation can be made.

Considering the model population parameters, the assumption that population size and selection pressure remain constant among generations, is unrealistic. In practice, these parameters will vary due to reasons as diverse as the availability of seed, availability of greenhouse space or field plots, compensation for the amount of genetic variance in the population, or, importantly, restrictions in personnel resources. The selection process in the model assumes no selective advantage among the selected individuals. All selected individuals contribute equally (subject to the random selection of the male parent in the case of outbreeding) to the following generation, whereas in reality factors such as degree of fertility and seed viability might have an effect.

The model presented here represents an improvement over earlier models in three respects: (1) additional features (e.g. partial allelic dominance and automatic repetition) have been incorporated; (2) whilst being suitable for use as an educational tool, the underlying mechanisms and the overall model design allow for the use and continued development of the model as a research device; (3) the physical structure of the model allows for the development and subsequent insertion of autonomous subroutines dealing with specific components of the selection process. In this respect, this model will be of potential use as a platform for continued use and development by others. The PSELECT program was made freely available to the academic community in May 1993 (Partner *et al.*, 1993). The

program represents an important intermediate stage in the development of a family of computer simulation models that, upon attainment of a sufficient level of sophistication and predictive accuracy, would be of practical use as decision-support and educational tools in the field of plant breeding.

III. Experimental Data for Comparison with Output from the PSELECT Simulation Model. Phenotypic Truncation Selection in Brassicas.

BREEDING OBJECTIVES

As part of the validation and verification procedures for the PSELECT computer model, it was decided to attempt to obtain experimental selection response data for comparison with output from the simulation model. Experimental populations of inbreeding and outbreeding representatives of the genera *Brassica* and *Sinapis* were subjected to phenotypic truncation selection with the aim of minimising plant height, whilst maintaining vigour and robustness. A range of parental crosses were made, selection in favour of shorter phenotypes was practised on the F_1 and subsequent generations, and phenotypic response to selection recorded. Breeding material was chosen to allow direct character comparison between morphologically similar inbreeder/outbreeder pairings.

MATERIALS AND METHODS

Species

(1) *Brassica carinata*, Braun. Ethiopian/Abyssinian mustard.

Little is known about *Brassica carinata*. Barring minor-scale research cultivation in Pakistan (M. Akhtar, personal communication), its cultivation seems to be limited to the Ethiopian plateau, where land races are used as a fodder crop and as a source of oil. A historical account of the species was given by Vaughan (1956), by which time the majority of research workers had placed it in the *Brassica juncea* complex. Burkill (1930) treated it as a variety of *B. juncea*, while Bailey (1930) was the first to accord it the separate species name, *Brassica carinata*.

Brassica carinata is an allotetraploid, its genomic complement, $\eta = (8+9) = 17$, being derived from the interspecific hybridisation of the diploids, *B. nigra*, $\eta = 8$, and *B. oleracea*, $\eta = 9$, (Morinaga, 1933; U, 1935; Mizushima, 1950; Harberd, 1972). There appears to be no knowledge regarding the history or origin of *B. carinata*. As the species is endemic of the Ethiopian plateau, Mizushima & Tsunoda (1967) undertook an unsuccessful search of the area for wild forms. They reported the occurrence of wild forms of *B. nigra*, which together with the reported ancient cultivation of the kale-like form of *B. oleracea* in the area, lends support to the theory that the area is the origin of genetic diversity for *B. carinata*.

The plant is an elegant and mechanically stable branched annual, with large, typically lobed leaves, which can take on a whitish appearance as a consequence of the expression of leaf waxes. Stem colour ranges from purple to pale green, and overall the plant is generally larger and sturdier than *B. juncea*. Multiple inflorescences bear small, green flower buds; fewer, larger and darker in colour than those of *B. juncea*. Flowers range from white to bright yellow in colour, and are of similar configuration to those of *B. juncea*. *B. carinata* is an outbreeder, but is tolerant of inbreeding. Seeds range from dark brown to black in colour, and are borne in long, laterally flattened pods. The internal structure of the pods is such that the seeds are arranged in two parallel columns, one either side of the pod's longditudinal centre line.

The *Brassica carinata* populations used in this experimental programme, were collected as seed in Ethiopia, and kindly provided by the Institute of Horticultural Research (now Horticulture Research International), Wellesbourne.

Population NVRS08/002485, "Addis Aceb"
Population IHRGRU08/004294A, "Chembere Dzagumhana"
Population NVRS08/004669, "Tamu Texel Greens"

(2) *Brassica junce*a (L.) Czernjaew, brown/Indian mustard.

Brassica juncea is a branched annual, with large, typically lobed leaves. Multiple inflorescences bear many small, light green flower buds, which open to reveal typically yellow flowers, with short, curved stamens. *B. juncea* is predominantly self-pollinating, although under field conditions approximately 30% cross-pollination can occur, depending on wind and insect activity (Rakow & Woods, 1987). Seed colour ranges from pale brown to black and seed is borne in semi-erect cylindrical pods. *B. juncea* is of significant economic importance, grown both as an oil crop and as a vegetable, and is also cultivated to a lesser extent for use as a condiment, to which it contributes pungency (Hemmingway, 1976). Different forms of the crop, cultivated for different purposes, display an almost unparalleled degree of morphological variation. Because of its high degree of polymorphism in leaf shape, the crop has been classified under various genera and species names. The extremely high level of morphological polymorphism within the species continues to give rise to contention regarding its full classification (Prakash & Hinata, 1980). For a comprehensive review of the taxonomy of *Brassica juncea*, refer to Vaughan *et al.* (1963).

Like *Brassica carinata*, *Brassica juncea* is an allotetraploid. Its genomic complement, η = (8+10) = 18, is believed to derive from the interspecific hybridisation of *Brassica nigra* and *Brassica campestris*, η = 10, (Morinaga, 1934; Sasaoka, 1930; U, 1935; Frandsen, 1943). More recent research has suggested that alternative η = 10 Brassica species, such as *Brassica japonica*, *Brassica pekinensis* and *Brassica trilocularis*, might have been the parents of *B. juncea* (Vaughan *et al.*, 1963), which would in part explain the observed level of polymorphism within the species. It should be pointed out, however, that most Brassica researchers place these η = 10 Brassicas as subspecies of *B. campestris*.

Prakash & Hinata (1980) stated that the earliest direct reference to *B. juncea* is found in ancient Sanskrit literature. The oldest physical evidence of the species comes in the form of seed remains excavated at Chanhu-Daro, India (Allchin, 1969). There is some disagreement regarding the centre of genetic diversity, but most researchers fall into one of two camps of thought: 1. that the species arose in central Asia, Western China (Sinskaia, 1928) and the North-Western Indian sub-continent (Vavilov, 1949); 2. that the species is of African origin, with subsequent secondary divergence from central Asia (Burkill, 1930; Zeven & Zhukovsky, 1975). Sun (1970) places some doubt on a Chinese origin due to the absence of ancestral forms of *B. nigra* and *B. campestris* in this area. Wild forms of *B. campestris* display a high level of morphological variation, with each variant having a characteristically limited geographical distribution. The observed polymorphism in *B. juncea* might therefore suggest that it arose independently in more than one area (Olsson, 1960; Vaughan *et al.*, 1963), wherever *B. nigra* and *B. campestris* coexist. The observation that wild forms of both *B. nigra* and *B. campestris*, together with wild forms of *B. juncea*, are found exclusively in Asia Minor and Southern Iran (Mizushima & Tsunoda, 1967; Tsunoda & Nishi, 1968) supports the hypothesis that a Middle-Eastern origin is likely.

The *Brassica juncea* lines used in this experimental programme were generously provided by Reckitt & Colman Ltd., Norwich.

Breeding line J/1078/5/4/89/2013, (unnamed).
Variety 89/2026, "Stoke"
Variety 89/2020, "Trowse"

(3) *Sinapis alba*, white mustard

Sinapis alba, also classified as *Brassica hirta* in North America, is a highly branched, seed-propagated annual, bearing large, typically bright green pinnately lobed leaves.

Inflorescences bear multiple buds, slightly fewer but larger than those of *B. juncea*. Flowers range from pale to bright yellow, and the uniformly pale yellow seed is borne in cylindrical pods. The crop is fast-growing and produces a great amount of foliage. *S. alba* is used primarily for the production of condiment, to which it contributes the "hot" principle (Hemmingway, 1976), as a break crop, and/or can be ploughed-in as a green manure.

S. alba is a diploid ($\eta = 12$), and a Mediterranean centre of genetic diversity has been postulated (Hemmingway, 1976). *S. alba* does not readily form interspecific hybrids, and only one instance of interspecific hybridisation with the crop Brassicas (*S. alba* x *B. oleracea*), has been reported (U, 1935). From this it can be established that there is no cross homology between the two genomes (Mizushima, 1950), and by extension that there is therefore little or no relationship with either *B. carinata* or *B. juncea*.

S. alba is a wind pollinated species which naturally exhibits sporophytic self-incompatibility common to many Brassicas (Hemmingway, 1976), although this self-incompatibility mechanism has been overcome, and *S. alba* is now propagated predominantly as fully inbred lines, such as those used in this programme.

The *Sinapis alba* lines used in this experimental programme were provided by Reckitt & Colman Ltd., Norwich.

Variety 560/A, "Kirby"
Variety 850/A, "Thorney"
Variety 990/B, "Tilney"

(4) *Sinapis arvensis*, Charlock or wild/field mustard

Sinapis arvensis is morphologically similar to, but commonly smaller than *S. alba*. The stem is generally hairy, and flower buds are invariably smaller than those of both *S. alba* and *B. juncea*. The seeds, borne in pods, are smaller than those of *S. alba*, and range from dark brown to black in colour. *S. arvensis* is a common weed of arable land throughout Europe (Fitter, 1978). Significant levels of seed dormancy are observed (Egley & Duke, 1985). *S. arvensis* populations "Boghall" and "Chambers" were harvested in Midlothian.

Parental crosses

Breeding lines were established by making the following reciprocal parental crosses:

Brassica carinata breeding lines:

CA. ADDIS ACEB pollinated by CHEMEBERE DZAGUMHANA
CB. CHEMBERE DZAGUMHANA pollinated by ADDIS ACEB
CC. ADDIS ACEB pollinated by TAMU TEXEL GREENS
CD. TAMU TEXEL GREENS pollinated by ADDIS ACEB
CH. CHEMBERE DZAGUMHANA pollinated by TAMU TEXEL GREENS
CK. TAMU TEXEL GREENS pollinated by CHEMBERE DZAGUMHANA

Brassica juncea breeding lines:

JA. J/1078/5/4/89/2013 pollinated by STOKE
JB. STOKE pollinated by J/1078/5/4/89/2013
JC. J/1078/5/4/89/2013 pollinated by TROWSE
JD. TROWSE pollinated by J/1078/5/4/89/2013
JH. STOKE pollinated by TROWSE
JK. TROWSE pollinated by STOKE

Sinapis alba breeding lines:

AA. KIRBY pollinated by TILNEY
AB. TILNEY pollinated by KIRBY
AC. KIRBY pollinated by THORNEY
AD. THORNEY pollinated by KIRBY
AH. TILNEY pollinated by THORNEY
AK. THORNEY pollinated by TILNEY

Sinapis arvensis breeding lines:

KA. BOGHALL pollinated by CHAMBERS
KB. CHAMBERS pollinated by BOGHALL

The containers in which the seed lines were kept were physically labelled with the appropriate two-letter breeding line code and the generation (taking the seed as the starting point of a generation). Great care was taken to ensure that seed lines were clearly labelled and kept separate at all times.

Sowing

Within constraints imposed by availability of greenhouse space, population size was maximised. After F_1 populations of 5 plants per cross, the F_2 population size was 15, increasing to 20 plants per cross for subsequent generations. Seeds were double-sown in 9" plastic pots, using the peat-based compost mixture described in the Appendix C1. Following germination and establishment, seedlings were thinned to one per pot. This operation was conducted when emerged seedlings were approximately 5 centimetres above-ground height. In cases where seeds did not germinate, excess seedlings were transplanted into the empty pots.

In order to ensure the requisite population size, 25 pots (18 for generation F_2) were sown for each breeding line. Sowing dates and approximate harvesting dates are given as Appendix C2. In the event that insufficient germinations were achieved, these extra plants could be substituted for the non-germinants. These extra plants were treated identically to the experimental population, and grown to maturity. In the event of excessive mortality within the experimental population, a plant from this additional group could be randomly selected for substitution into the experimental group. It should be noted, however, that these additional plants would only be used in the event of the population size falling below 20 prior to measurement and pollination.

Individual plants were labelled with the following information:

(a) *Generation*
(b) *Population/breeding line*
(c) *Plant identification number*

So, for example, an individual plant would bear the label, **F_3 - CK - 17** to denote plant 17, of breeding line CK, in generation F_3.

Most plants required staking and loose tying upon attainment of a height of approximately 50cm. In most cases, this was a precautionary measure. In field populations, the plants

were seen to attain a greater level of mechanical strength, and structural stability, and staking was therefore not employed, except in cases of obvious mechanical deficiency.

Measurements

Plants were grown to first flowering, taken to be the time at which the first flower bud opens to reveal the flower petal. Time to first flowering will therefore be different for each plant within the population, and can be expected to vary among generations as a consequence of different environmental conditions.

At first flowering, the following characters were scored for each plant:

Objective measurements:

L1-F: The length from the first leaf-node above ground, to the base of the flower head. Measurement in millimetres, but rounded to nearest 5mm interval.

N1-F: The number of leaf-nodes inclusive of the first node above ground, to the flower.

Subjective measurements:

DAD: (Degree of apical dominance) A subjective appraisal of degree of branching. Score ranges from 1 (high degree of apical dominance, low branching) to 5 (low degree of apical dominance, highly branched).

ADL: (Apical dominance lost) The leaf-node, inclusive of the first node above ground, at which the main vertical growth of the plant can no longer be unambiguously identified. Scored as a negative value relative to the flower head. Thus a larger distance from the apical meristem is denoted by a larger deviation from 0.

StT: A subjective assessment of stem thickness. A score of 1 represents an unusually thin stem, while a score of 3 represents an unusually thick stem.

For subjective measurements it was important to gain an appreciation of what constituted the "typical" or "average" stem thickness within the population. Plants having a significantly thicker stem would score 3 for stem thickness, whilst significantly thin, weak-looking plants would score 1 and be eliminated from the breeding programme. An equally

valid nomenclature for these subjectively assessed parameters would be to score characters as "+" or "-" with respect to the typical value. A numerical classification was adopted for ease of data handling.

Selection

The breeding objectives can be summarised as minimising L1-F, whilst maintaining a vertical growth habit, and sufficient stem thickness to confer mechanical stability to the plant. N1-F was not subjected to active selection, but was monitored to observe the secondary effects of selection on L1-F, primarily to determine whether a reduced plant height was conferred by a reduction in the number of leaf nodes, by a reduction in length of the internodes, or by a combination of both factors. Individuals scoring DAD scores of 4 and 5, and a StT score of 1, were defined as lethal, and excluded from taking further part in the selection programme, irrespective of their scores for L1-F and N1-F.

Phenotypic proportional truncation selection was effected upon the experimental populations. The "best" (those with smallest L1-F scores) 20% of each population was selected for onward progression. e.g. with a population size of 20, the best 4 were selected. All other plants were discarded upon completion of all measurements.

Pollination

(1) Outcrossing *Brassica carinata* and *Sinapis arvensis* populations, CA, CB, CC, CD, CH & CK; KA & KB. The prime objective was to achieve random cross-pollination among the selected individuals.

The pollinator (male)

Upon flowering, pollen was collected from mature flowers (taken to be those in which the anthers had taken on a "hairy" appearance as a result of surface pollen) of the pollinator plant. The anthers (in reality, the whole flower) were removed and stored in dry, sterile, labelled petri dishes. Four pollen pools were formed from the four "best" phenotypes. Anthers were removed, and for each selected plant, anthers were put into 3 out of 4 labelled petri dishes. The pollen pools so formed were then used to pollinate the selected plants, with each plant being pollinated using the pollen pool to which it had not contributed pollen. In this way, outcrossing among the selected parents was ensured. A diagrammatic representation of this pollination scheme is presented as Figure 3.1.

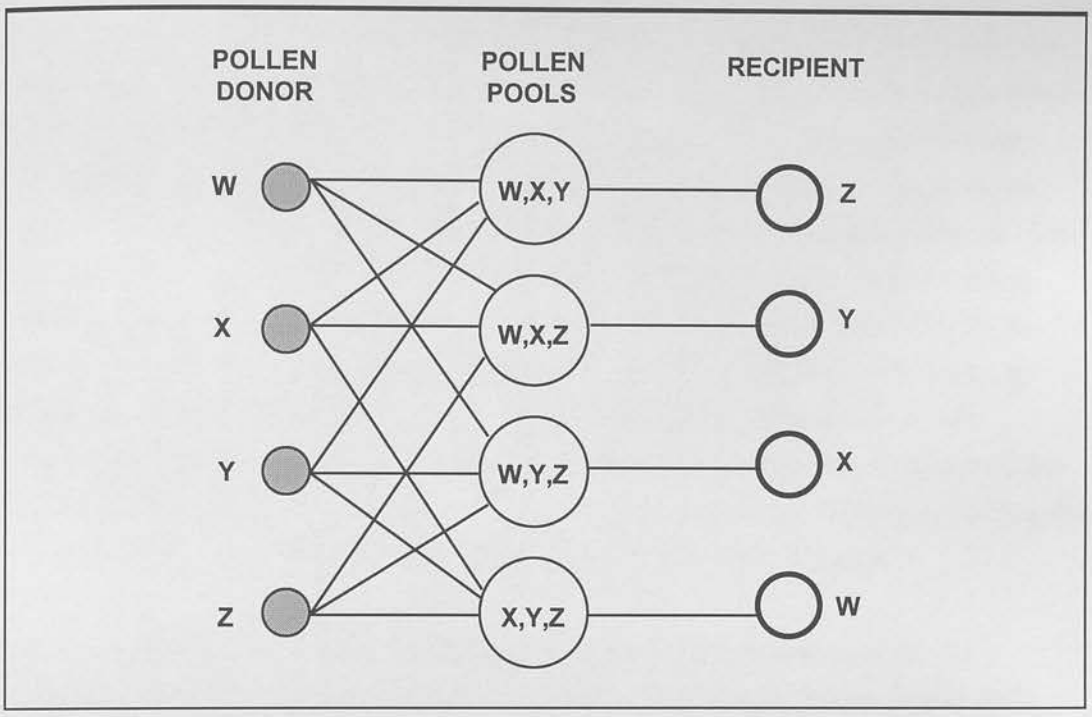


Figure 3.1. Pollination protocol for outbreeding populations.

The recipient (female)

On the female parent, flower heads bearing a high number of flower buds at the appropriate stage of development, generally 1-2 days prior to opening, were chosen for emasculation. All open flowers and immature buds were removed with fine scissors, and the sepals, petals and anthers of the remaining buds were removed using fine forceps, to expose the stigma.

Firstly, the flower head chosen for emasculation was prepared by removing unwanted side flower buds, to leave the desired flower head at least 3 to 4 centimetres clear from its nearest remaining neighbour. Care must be taken at this stage to avoid causing damage to the plant stem. Vaseline jelly was applied to the sites of bud removal, to resist desiccation, and to provide a physical barrier against pathogen entry.

Those flower buds too small for emasculation (immature buds are generally concentrated towards the centre of the flower head) were removed using fine scissors. Due to the destructive nature of the emasculation process, and the resultant level of bud wastage, great care was taken to remove only those buds that were too small.

Once superfluous buds had been removed, each remaining flower bud was taken in turn, and the outer sepals, petals and anthers, were removed to expose the stigma. When all flower buds were thus prepared, a 3mm paintbrush was employed to act as the agent of pollination. Pollen from the appropriate pollen pool was gently dusted onto the exposed ends of the stigmas to effect pollination. The stigma are highly prone to mechanical damage at this stage, and after experimentation, it was ascertained that the best method of pollination was to avoid physical contact between the brush and stigma. By holding the pollen-bearing brush approximately 0.5 to 1 centimetre above the exposed stigma, and then flicking the stem of the brush, enough pollen was ejected from the brush bristles onto the stigma, to effect pollination. It should be noted that the majority of the pollen is wasted in this manner, but as the amount of pollen was not limiting in this experimental programme, this was not an important consideration. In situations where the amount of pollen is limiting, the deliberate physical placement of the pollen onto the stigma might be desirable.

In order to prevent undesired cross-pollinations, all instruments were sterilised (by immersion in 97% v/v ethanol), and dried between using each pollen pool. A dedicated brush was used for each pollen pool.

Following pollination, the pollinated flower heads were isolated using a hybridisation bag. In the species in this programme, 90mm x 275mm photographic negative bags were found to be satisfactory for use in this capacity. Because those populations grown in the glasshouse were not subjected to wind, it was possible to leave the bags relatively loose on the flower head. This is desirable to prevent mechanical restriction to growth as the flower head matures. For field populations, the hybridisation bag must be fastened at the base (commonly with a staple).

All pollinated flower head were then labelled with a small parcel tag, immediately above the first branch point below the bottom edge of the hybridisation bag, so that everything above the label was contained within the hybridisation bag, and hence isolated. In harvesting, nothing was taken from below this label.

Bags were checked routinely to ensure that the physical encapsulation of the flower heads was causing no mechanical damage to the plant. It was essential to prevent the bag from physically restricting the growth of the plant, and great attention was required to keep the bag securely but loosely fastened over the exposed regions. Hybridisation bags were removed upon set of the siliquas. Label tags were left in place for the duration of the experiment.

(2) Self-pollinating *Brassica juncea* and *Sinapis alba* lines, JA, JB, JC, JD, JH & JK; AA, AB, AC, AD, AH & AK.

The four best phenotypes were selected and labelled with tags. Self-pollination was effected by isolating labelled flower heads within hybridisation bags. Subsequent experimental procedures were as described for *B. carinata*. Again, hybridisation bags were removed upon set of the siliquas. Tags were left in place.

Selected plants were grown to maturity, whereupon the seed from the labelled flower heads was harvested. Harvested seed was mixed in equal proportions (e.g. 1g of seed from each plant) to establish the seed stock to be resown as the subsequent generation, F_{t+1}.

RESULTS & ANALYSIS

Results are presented for *Brassica carinata* and *Brassica juncea* crosses. Unavailability of sufficient greenhouse resources meant that the scale of experimentation had to be drastically reduced. Germination of *S. arvensis* was consistently poor. As the diploid outbreeder to complement *S. alba*, this led to the inevitable decision to sacrifice the range of populations under study, rather than reducing population size. *Sinapis alba* lines were discontinued prior to the F_3 stage. Due to this early discontinuation of the *Sinapis* breeding programme, these data are not presented. *Sinapis* data have been retained on file. Population figures for *Brassica* spp. crosses are presented as Appendix C3.

Cytoplasmic effects

Reciprocal crosses were made as described in the materials and methods section. Response of population mean plant height (L1-F) to selection was plotted for *B. carinata* (Figure 3.2a) and *B. juncea* (Figure 3.2b). Noticeable deviation between F_1 L1-F scores for reciprocal crosses would suggest the operation of cytoplasmic factors. On the basis of these plots, cytoplasmic effect was qualitatively discounted for both *B. carinata* and *B. juncea*. A single-factor analysis of variance (ANOVA) test was conducted on reciprocal F_1 populations. Results of the ANOVA are presented as Appendix C4. Only one cross (*B. juncea* breeding lines JH/JK) showed significant (at 5% level) difference between reciprocals. In the absence of significance in related crosses, this result was attributed to small population size (insufficient for a reliable estimate of background variation), and discounted.

Response to selection

Summary statistics for all *Brassica* spp. crosses were used to plot response of L1-F and N1-F to phenotypic truncation selection for reduced plant height (Figures 3.3 to 3.7). Positive (downwards) response to selection was observed in all breeding lines. Environmental variation clearly had an effect, as F_3 values of L1-F and N1-F for all breeding lines (with the exception of *B. carinata* lines CH, CK and *B. juncea* line JD) displayed an abnormally large positive deviation from surrounding values. Generation F_4 showed a severe drop in values (probably attributable to high F_3 scores) for all breeding lines (moderate drop in line CH).

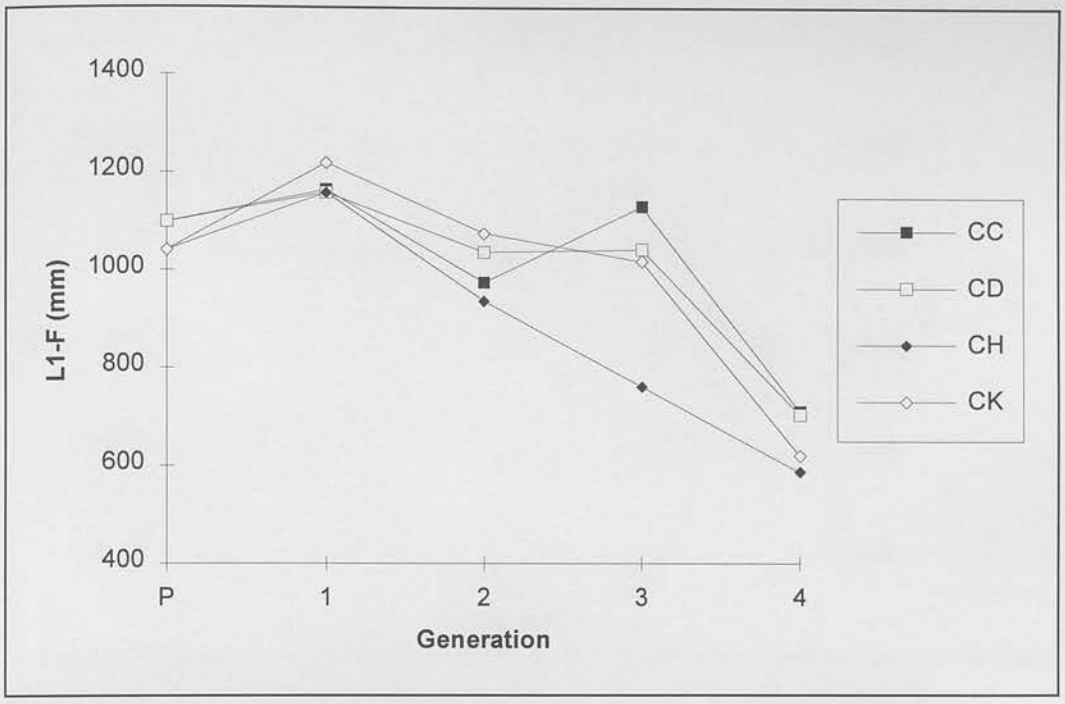


Figure 3.2a. Response of plant height (L1-F) to downwards selection, *Brassica carinata* breeding lines CC, CD, CH and CK.

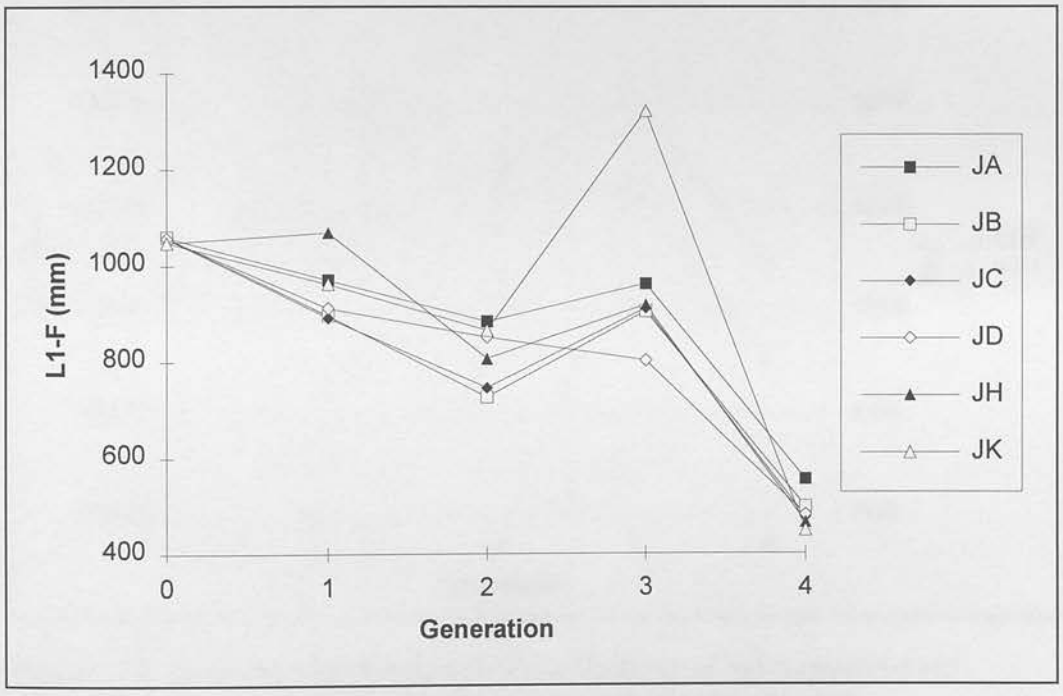


Figure 3.2b. Response of plant height (L1-F) to downwards selection, *Brassica juncea* breeding lines JA, JB, JC, JD, JH and JK.

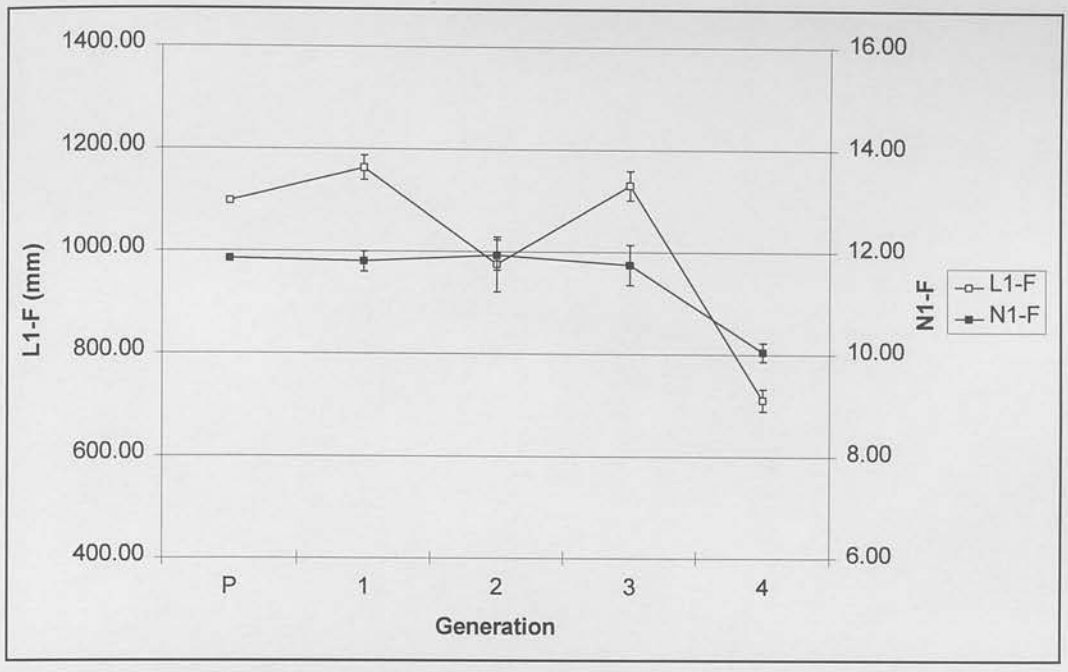


Figure 3.3a. Response of plant height (L1-F) and number of leaf nodes (N1-F) to downwards selection on plant height, *Brassica carinata* breeding line CC.

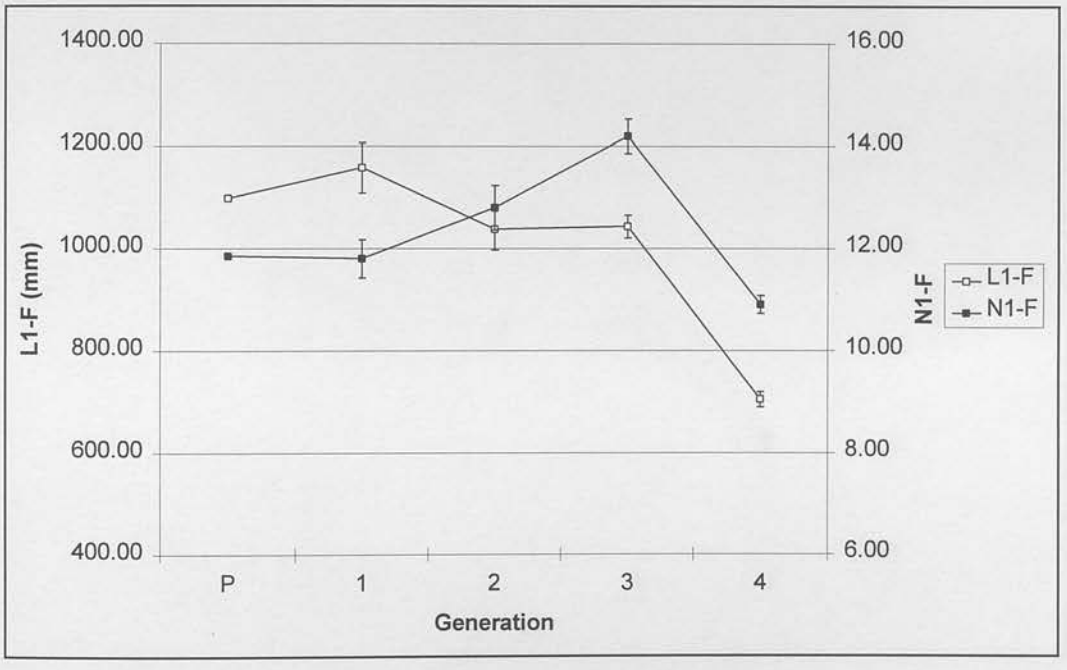


Figure 3.3b. Response of plant height (L1-F) and number of leaf nodes (N1-F) to downwards selection on plant height, *Brassica carinata* breeding line CD.

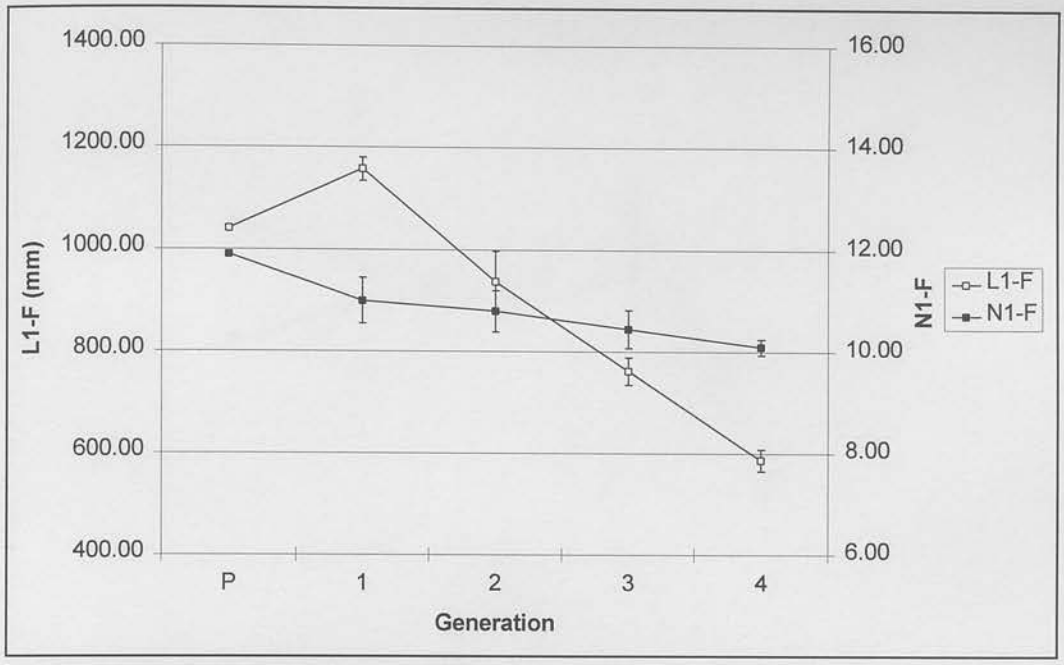


Figure 3.4a. Response of plant height (L1-F) and number of leaf nodes (N1-F) to downwards selection on plant height, *Brassica carinata* breeding line CH.

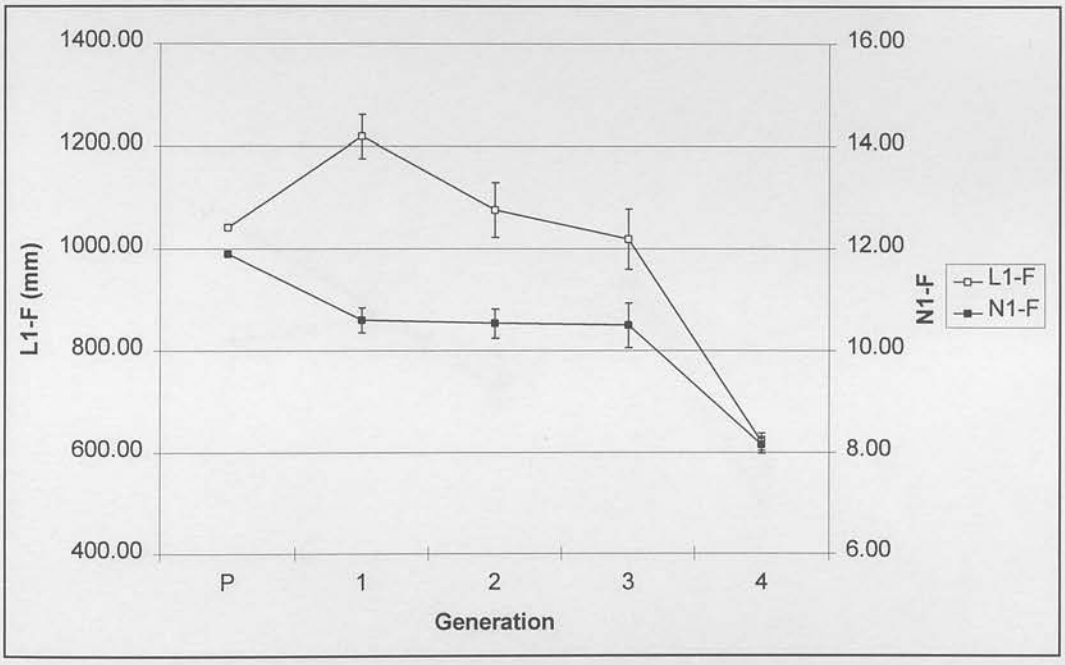


Figure 3.4b. Response of plant height (L1-F) and number of leaf nodes (N1-F) to downwards selection on plant height, *Brassica carinata* breeding line CK.

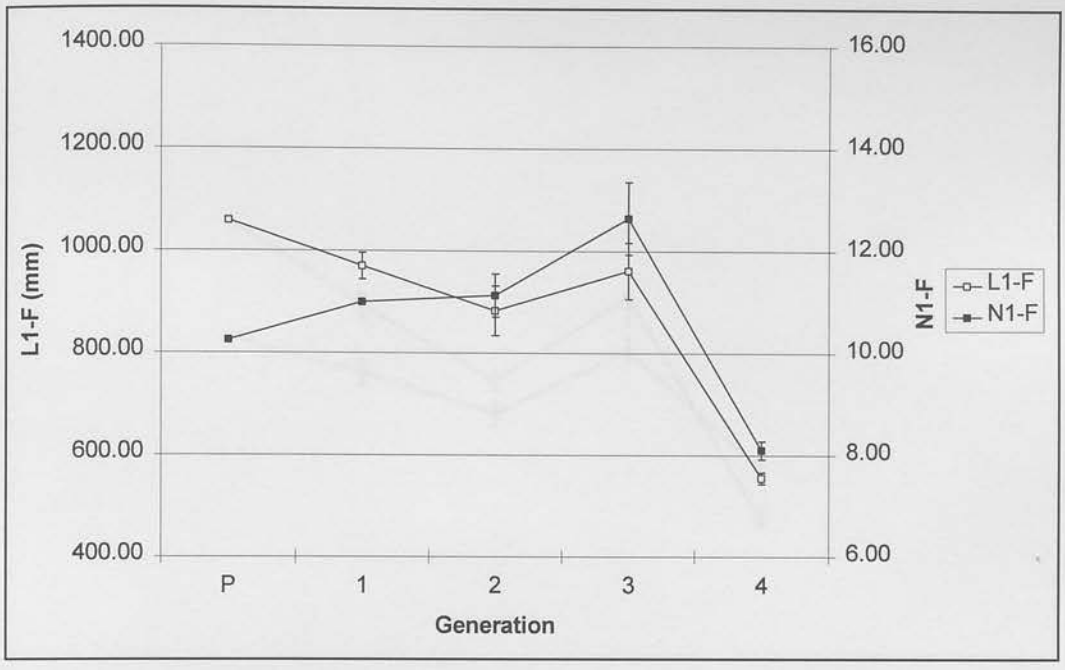


Figure 3.5a. Response of plant height (L1-F) and number of leaf nodes (N1-F) to downwards selection on plant height, *Brassica juncea* breeding line JA.

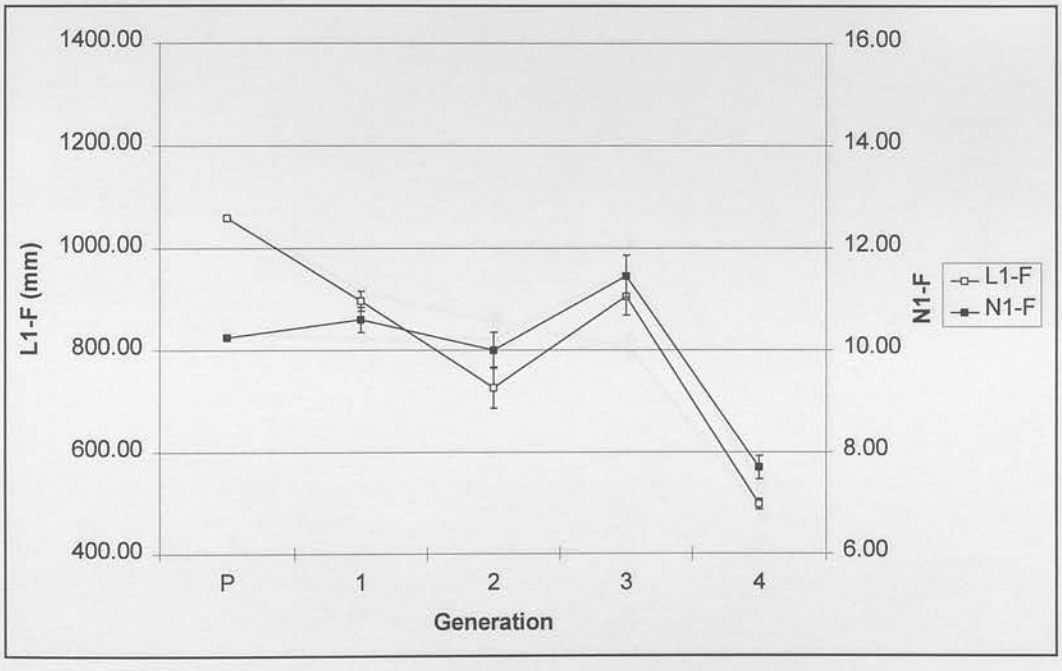


Figure 3.5b. Response of plant height (L1-F) and number of leaf nodes (N1-F) to downwards selection on plant height, *Brassica juncea* breeding line JB.

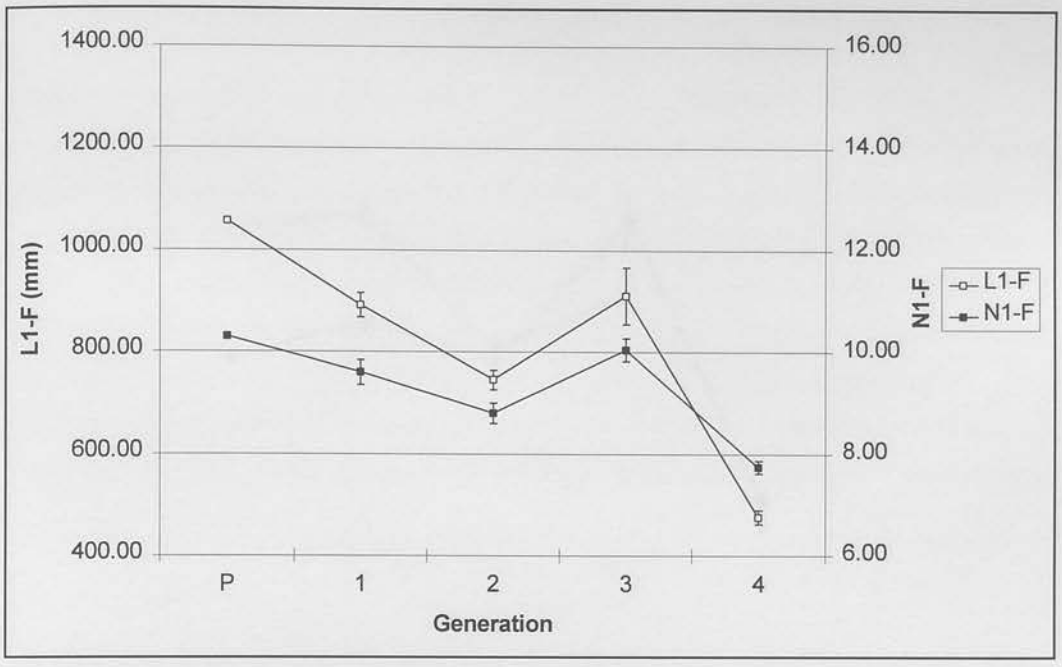


Figure 3.6a. Response of plant height (L1-F) and number of leaf nodes (N1-F) to downwards selection on plant height, *Brassica juncea* breeding line JC.

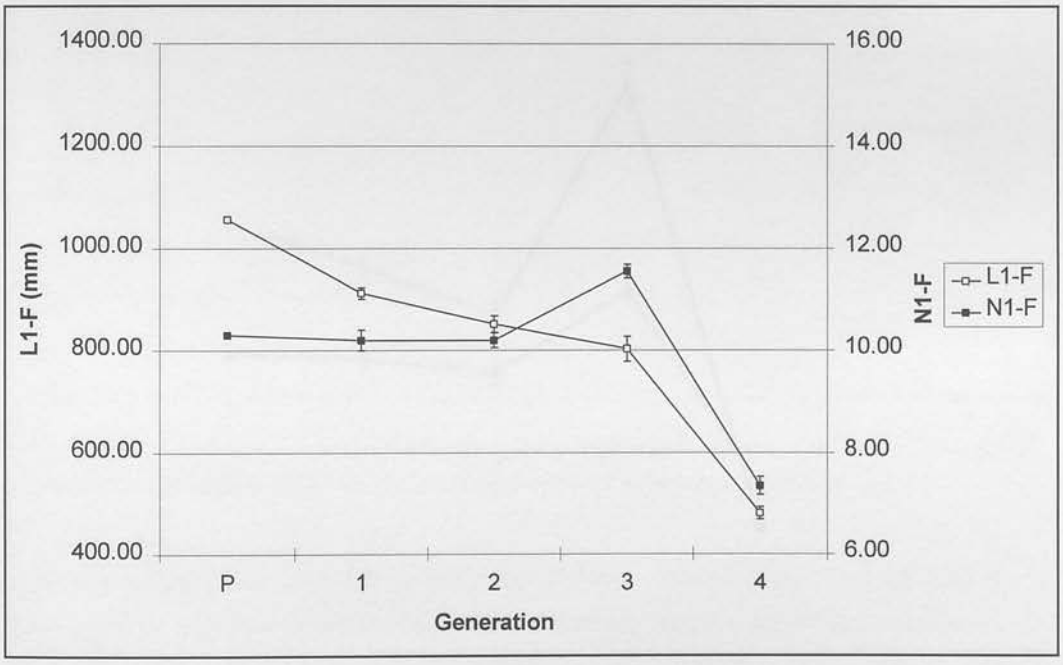


Figure 3.6b. Response of plant height (L1-F) and number of leaf nodes (N1-F) to downwards selection on plant height, *Brassica juncea* breeding line JD.

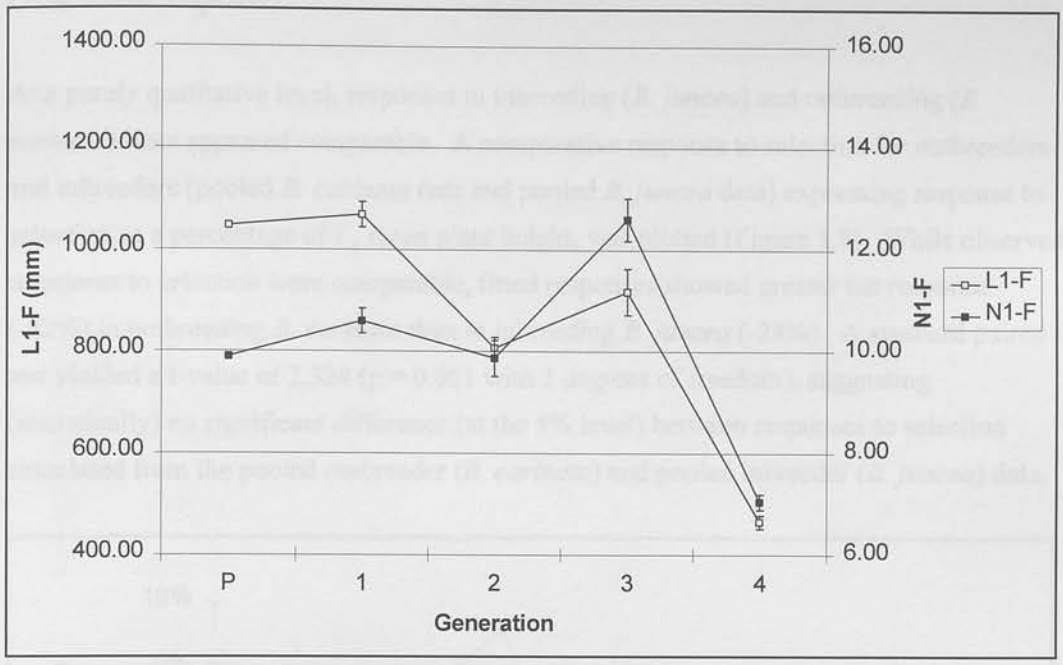


Figure 3.7a. Response of plant height (L1-F) and number of leaf nodes (N1-F) to downwards selection on plant height, *Brassica juncea* breeding line JH.

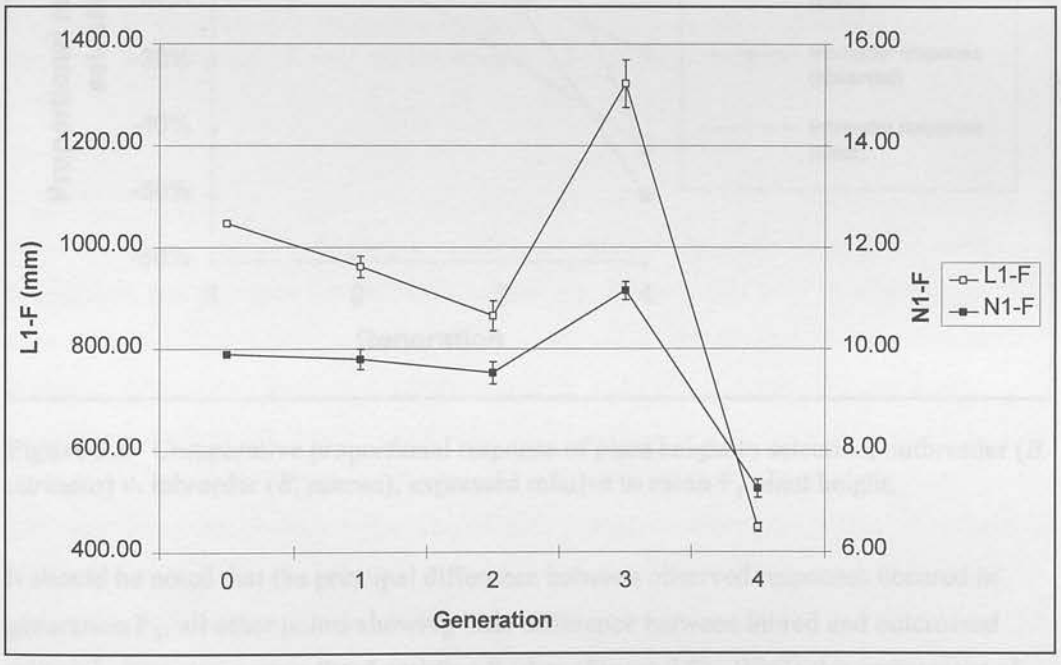


Figure 3.7b. Response of plant height (L1-F) and number of leaf nodes (N1-F) to downwards selection on plant height, *Brassica juncea* breeding line JK.

Comparative response

At a purely qualitative level, responses in inbreeding (*B. juncea*) and outbreeding (*B. carinata*) lines appeared comparable. A comparative response to selection for outbreeders and inbreeders (pooled *B. carinata* data and pooled *B. juncea* data) expressing response to selection as a percentage of F_1 mean plant height, was plotted (Figure 3.8). While observed responses to selection were comparable, fitted responses showed greater net response (-42%) in outbreeding *B. carinata* than in inbreeding *B. juncea* (-28%). A standard paired t-test yielded a t-value of 2.324 ($p = 0.051$ with 3 degrees of freedom), suggesting (statistically) no significant difference (at the 5% level) between responses to selection calculated from the pooled outbreeder (*B. carinata*) and pooled inbreeder (*B. juncea*) data.

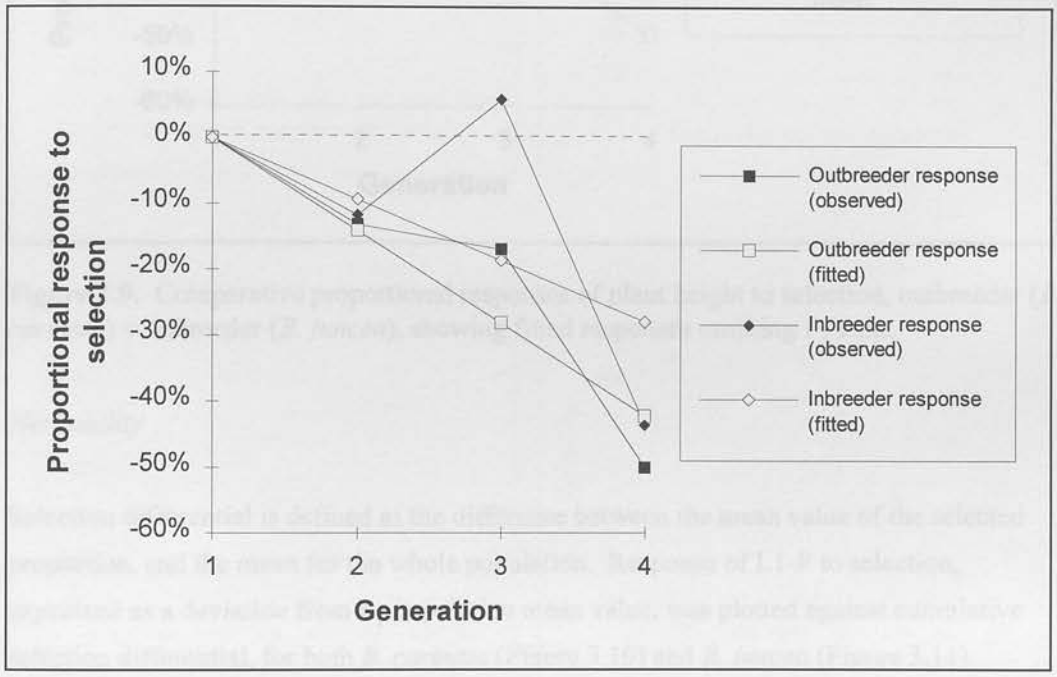


Figure 3.8. Comparative proportional response of plant height to selection, outbreeder (*B. carinata*) v. inbreeder (*B. juncea*), expressed relative to mean F_1 plant height.

It should be noted that the principal difference between observed responses occured in generation F_3, all other points showing little difference between inbred and outcrossed material. Responses were fitted omitting F_3 data (Figure 3.9). While this assessment of response to selection is unreliable by virtue of subjectively omitted data, and by virtue of having fitted a regression using only 3 data points (one of which, the origin, was fixed), it results in a more accurate description of comparative response. Fitted responses (ex-F_3)

showed comparable net responses to selection for *B.carinata* (-49%) and *B. juncea* (-43%), and were comfortably not significantly different ($p = 0.135$) at the 5% level.

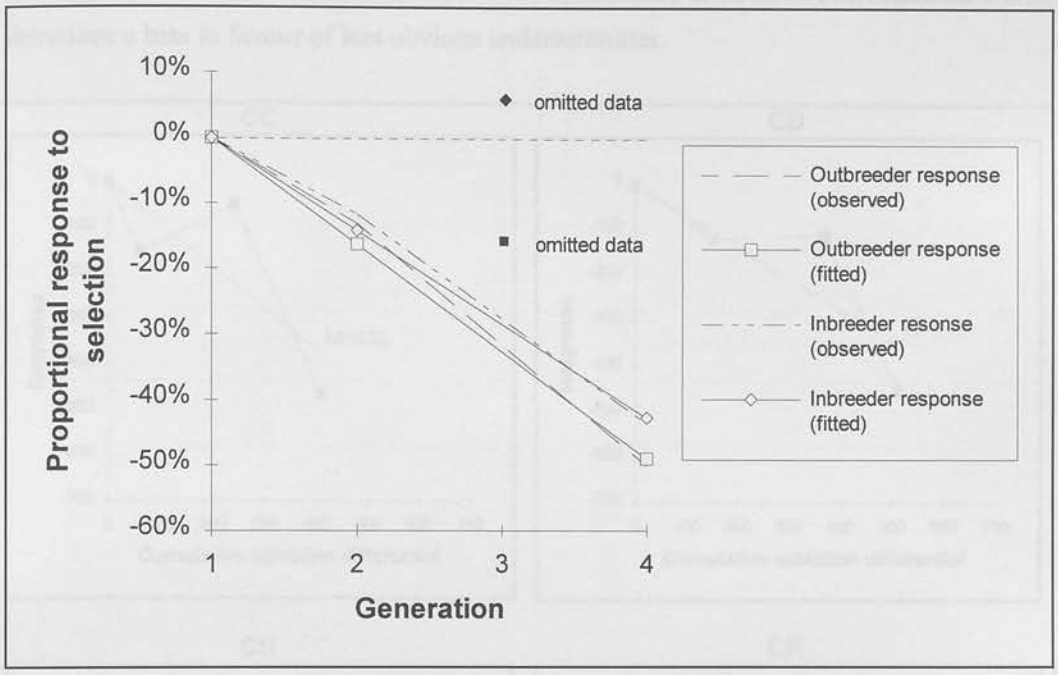


Figure 3.9. Comparative proportional responses of plant height to selection, outbreeder (*B. carinata*) v. inbreeder (*B. juncea*), showing fitted responses omitting F_3 data.

Heritability

Selection differential is defined as the difference between the mean value of the selected proportion, and the mean for the whole population. Response of L1-F to selection, expressed as a deviation from F_1 population mean value, was plotted against cumulative selection differential, for both *B. carinata* (Figure 3.10) and *B. juncea* (Figure 3.11). Regression analyses of observed response against cumulative selection differential yielded estimates of the realised heritability (h^2) of L1-F, given by the gradient of the fitted response. Data from which realised heritabilities were calculated are presented as Appendix C5. Estimates of h^2 varied from 0.69 to 1.37, with a mean of 0.97 (s.e.m. = 0.143), for outbreeding (*B. carinata*) populations, and from 0.55 to 1.91, with a mean of 0.98 (s.e.m. = 0.227), for inbreeding (*B. juncea*) lines. The similarity between these estimates is in keeping with the observation of equivalent response to selection for inbreeders and outbreeders.

It should be noted that heritability, by definition, can have values from 0 to 1. Estimated heritabilities exceeding 1 are therefore inaccurate. Such experimentally-derived estimates

should, however, be considered when calculating a mean realised heritability (D.S. Falconer, personal communication), as in the absence of experimental bias any inaccuracy in estimation would not be directional, and the elimination of obvious overestimates would introduce a bias in favour of less obvious underestimates.

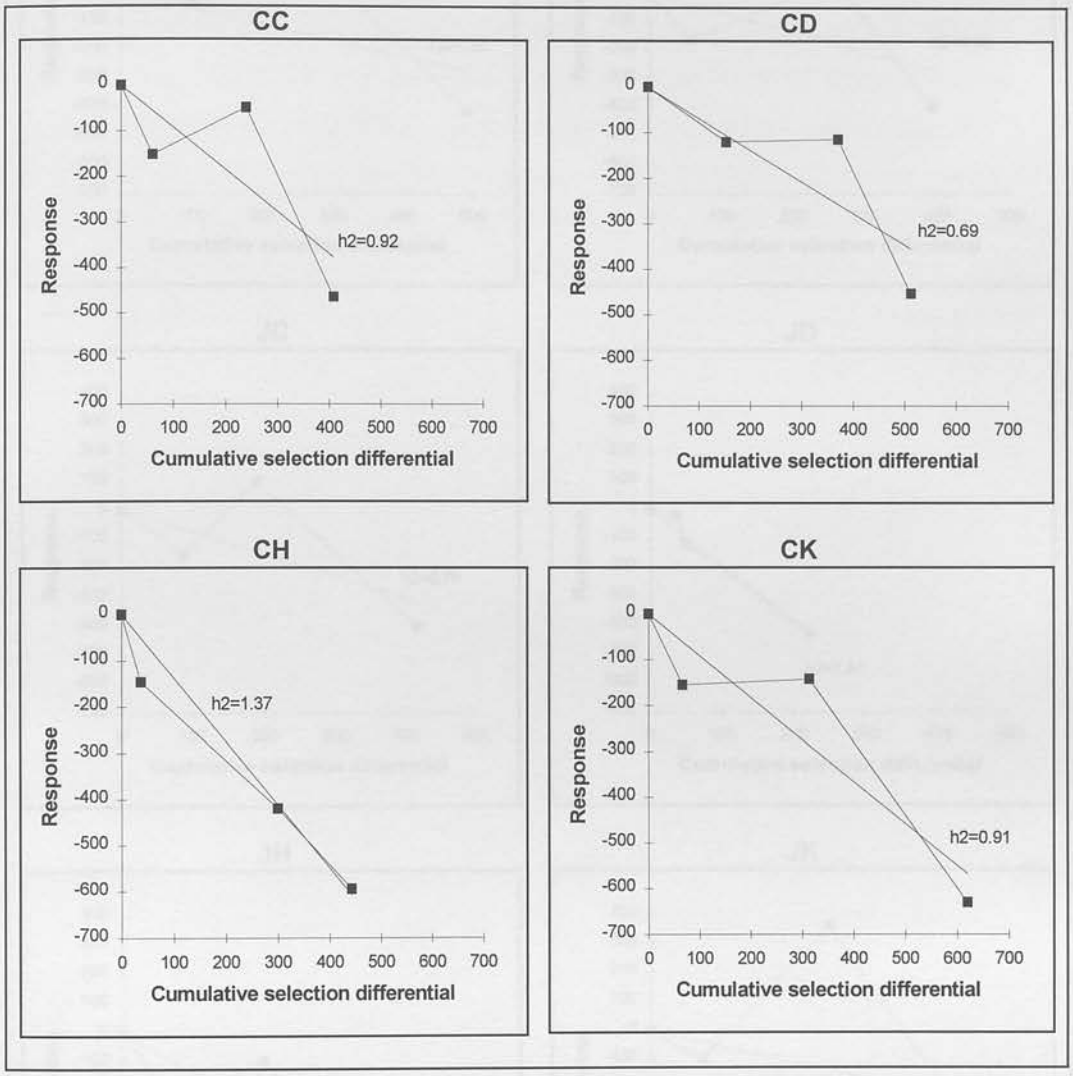


Figure 3.10. Plots of observed response (▪) to selection (expressed as deviation from F_1 mean) against cumulative selection differential, yielding estimates of realised heritability for L1-F in outcrossed *B. carinata* from the fitted response (—).

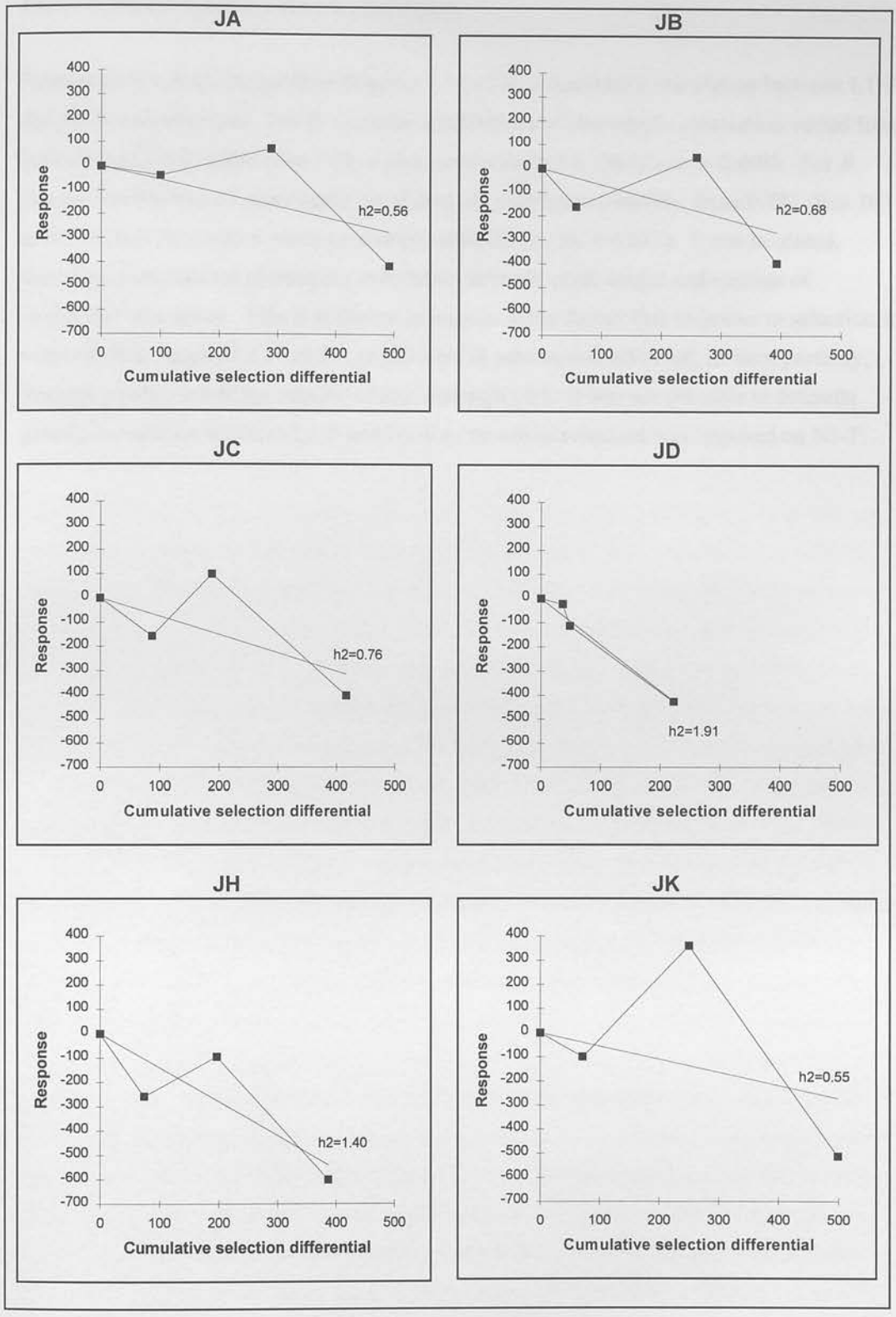


Figure 3.11. Plots of observed response (▪) to selection (expressed as deviation from F_1 mean) against cumulative selection differential, yielding estimates of realised heritability for L1-F in inbred *B. juncea* from the fitted response (—).

Phenotypic correlation between characters

From selection response profiles (Figures 3.3 to 3.7) a qualitative correlation between L1-F and N1-F was observed. For *B. carinata*, coefficients of phenotypic correlation varied from 0.441 (line CD) to 0.890 (line CC), with a mean value of 0.726 (s.e.m. = 0.098). For *B. juncea*, coefficients of phenotypic correlation showed less variability, from 0.731 (line JH) to 0.990 (line JK), with a mean correlation of 0.832 (s.e.m. = 0.051). It can be stated, therefore, that a strong phenotypic correlation between plant height and number of internodes was noted. This is evidence in support of the theory that response to selection for reduced plant height (L1-F) in *B. carinata* and *B. juncea* was achieved, as least partially, through a reduction in the number of leaf nodes (N1-F). It was not possible to estimate genetic correlation between L1-F and N1-F as no active selection was imposed on N1-F.

DISCUSSION

Experimental considerations

Mustards were chosen because of the relatively rapid generation interval, suitability for glasshouse cultivation in pots, and because the required range of genetic systems was represented. Due to the time constraints imposed upon the project, it was necessary to grow the populations in an uninterrupted tandem series, irrespective of seasonal factors such as light intensity or temperature. Attempt was made to keep these factors as constant as possible through the use of artificial lighting and the regulation of greenhouse ventilation, but it must be appreciated that populations grown over the winter months will show a different rate and pattern of growth to those grown in the summer.

Sinapis arvensis proved extremely difficult to cultivate in pots. All attempts at glasshouse cultivation failed, as did initial attempts in an outdoor cage. Minor success was achieved using a controlled environment growth chamber, but unavailability of this resource meant that this matter could not be pursued further. Given that charlock is a widely encountered weed species (Fitter, 1978), it is ironic that this should be the material with which most difficulty was experienced. Germination was erratic, which suggests the operation of a seed dormancy factor. As an additional experimental procedure, *S. arvensis* seeds were chitted with a range of gibberellic acid preparations. Best results were achieved by scratching the seed coats, and germinating using 10ppm gibberellic acid for 2 days at 20°C. In a number of repetitions, seedlings, without exception, died shortly after germination. In light of the failure to establish *S. arvensis* lines, the decision to discontinue *S. alba* lines was inevitable. *Brassica carinata* crosses ADDIS ACEB x CHEMBERE DZAGUMHANA repeatedly failed to germinate. In the absence of information to the contrary, one can only assume that this cross was inviable.

Environmental variation could not be eradicated from the experimental protocol, and for this reason the morphological characters under selection were assessed at a physiologically determined point, rather than with reference to time. First flowering was chosen as the point at which the selected characters were assessed as the environmental factors that affect flowering would be constant across the individuals that constitute the generation. Time to flowering, in days, would be an ideal character upon which to effect selection, but as this would be expected to be heavily influenced by environmental factors, selection for this character would only be reasonable in the absence of environmental variation, such as could be established in a growth chamber. The size and number of the experimental material,

combined with the expense and unavailability of such a resource precluded meaningful assessment of this character. By a similar argument, recordings of morphological characters taken at a pre-defined time, for example 60 days post germination, would be inappropriate. The developmental stage of the population would be expected to vary with seasonal changes in growth environment, and would therefore not provide a true characterisation of the population.

The main disadvantage of growing mustards in a greenhouse is the size of the mature plants. Population size was constrained by the physical limitations of available space. *Sinapis alba* lines were discontinued prior to the F_3 stage. Seed of *Brassica nigra* (black mustard), a diploid outbreeder, later became available. This could have been used as the complementary species to *S. alba*, but within the time limitations of the research, there would have been insufficient time to develop the *B. nigra* populations to an advanced stage. It was felt that concentration on those species well suited to greenhouse cultivation would yield better results.

The pollination protocol for outbreeders assumes an equal pollen contribution from the pooled pollen of the pollen donors. Within the scope of this investigation there is no way to test the validity of this assumption. There is also the inherent assumption of equal viability and reproductive fitness among gametes. It would not be unreasonable to expect selection at the gametic level, with pollen from the respective donors showing differential efficiencies of pollination. A final assumption applicable to the adopted selection protocol is that of phenotypic indistinguishability among offspring from gametes (environmental masking). This would be effective at two stages. Firstly, no account was taken of difference among seed produced by a given selected plant. Recombinations can be expected to generate variation among seeds. Selection, all be it unintentional and one assumes undirectional, is effected by the simple action of sowing a seed sampled (randomly) from a pooled source (pooled from pods of a given plant, and then pooled again among plants). The offspring (seed) population can therefore be expected to be comprised of a number of sub-sets, with seed from pods of the same plant being more genetically alike than those from sister plants.

The breeding programme was discontinued after generation F_4 due to constraints of time and resources. With the benefit of hindsight, the experimental procedure was flawed. The original aim of the selection was to conduct an experimental-scale breeding programme which as closely as possible mirrored typical practices of commercial breeding operations, where the primary aims are to maximise population sizes (allowing for a higher intensity of selection) and minimise generation interval (in order to progress through as many

generations as possible in as little time as possible). This is perfectly appropriate if the objective of the selection programme is to produce a finished variety in the shortest possible time. Given that the objective of this programme was to generate data suitable for validation of PSELECT output, the failure to take account of the contribution of environment to phenotypic variance is serious. In this case, it might have been appropriate to maintain control populations (to which no directional selection would be applied), despite the detrimental effect this would have on sizes of the selected populations. The measured observation would be the phenotypic difference between selected and control means.

Whilst undeniably having serious flaws, the experimental protocol adopted had some positive aspects. Were control populations maintained, population sizes would be halved for each cross. This would give rise to two (rather than one) sources of sampling variance (one per population), and the relative contribution of error variance to the overall variance within each population would be greater (D.S. Falconer, personal communication). In population sizes of 10, this can be expected to be significant. The failure to maintain control populations is therefore, at least to some extent, justifiable in terms of introduced error variance.

Due to the requirements of time, the breeding programme had to be initiated at the soonest opportunity, which in this case meant that the selection experiment was started fully 3 years prior to the first successful trial of the PSELECT model, which incidentally coincided with the discontinuation of the selection experiment. That there is some discrepancy between the nature of the data yielded by the selection programme, and the ideal specification for model validation data, is perhaps unsurprising.

Resources were not available to conduct a second breeding programme, although some work was subsequently conducted using Rapid-Cycling Brassica (RCB) material (Williams & Hill, 1986), selecting for reduced number of days to flowering in a controlled environment growth-room. F_2 RCB seed was harvested, but continued monopolisation of the growth-room facility was unreasonable, resulting in the discontinuation of this programme prior to obtaining a selection response.

Analysis

Any cytoplasmic (maternal) factors influencing response of plant height to selection, would be revealed in the F_1 generation. Cytoplasmic inheritance with respect to L1-F was discounted. In the absence of cytoplasmic effects, data from reciprocal crosses could be

pooled to increase the population size of a given cross. If data were pooled in this way, selection would no longer be by truncation, as phenotypic truncation selection would be effected on independent subsets of the population resulting from a given cross. Despite discounting cytoplasmic effects, data from reciprocal crosses was not pooled. In principle, the maintenance of reciprocal populations (in the absence of cytoplasmic effects) constitutes a replicated selection. It would therefore be possible to calculate the standard error of realised heritability for the L1-F character. This would, however, have little value with only two replicates.

It was noted that, for most crosses, generation F_3 showed a noticeable deviation from the general response. This deviation was attributed to environmental effects. F_3 data was included in the estimation of the realised heritability of L1-F from the regression of phenotypic response to selection against cumulative selection differential. It would also be possible to plot a second regression, omitting F_3 data. This second estimate of realised heritability would in all likelihood be more accurate (but less reliable by virtue of missing data) than the first. Given the stated aims of this breeding programme, it was felt that little benefit would be obtained from additional estimates of L1-F realised heritability. In addition to this, given the small population sizes employed, genetic drift can be expected to have made a significant contribution to the standard error of the estimated heritabilities (D.S. Falconer, personal communication). It would be unwise, therefore, to place too much importance on the estimates of h^2 obtained. It should also be noted that any estimate of heritability is unique, applying only to a specific population in a specific experimental context.

As previously stated, estimated realised heritabilities exceeding 1 are, by definition, incorrect. In circumstances where $h^2 > 1$, it can be assumed that greater phenotypic response to selection than expectation is a consequence of environmental contribution. It can be assumed that the calculated mean heritabilities of L1-F for *B. carinata* ($h^2 = 0.97$) and *B. juncea* ($h^2 = 0.98$) are overestimates, as further evidenced by the observed effect of environment on the breeding material. Estimated heritabilities, minus 1 s.e.m., yield heritability values of 0.83 and 0.75 for *B. carinata* and *B. juncea* respectively. These are roughly in line with experimental estimates for non-fitness characters (for example stature in man, $h^2 = 0.65$ (Roberts *et al.*, 1978); oil content in sunflower, $h^2 = 0.72$ (Fick, 1975)) and are likely to be better approximations to the "true" heritability of the L1-F character. Selection upon fitness-related characters typically gives lower realised heritabilities, typically in the range 0.10 to 0.30 (Falconer, 1981). It is not possible to comment on

observed vs. predicted responses to the imposed pressure of selection, as the heritabilities of plant height characters in Brassicas have not previously been assessed.

For the sake of experimental integrity, all calculations involving selection differential were performed using the mean L1-F of "surviving" members of the population (defined as individuals with selection character scores, DAD < 4 and StT >1), irrespective of physical survival or otherwise. A consequence of this is that population size, and hence proportion of the population selected, would not be constant across generations. This can be expected to result in discrepancy between results from this breeding programme and output from the PSELECT simulation model, where the assumption of constant population size is made. Having acknowledged this, observed differences between population mean L1-F values including and excluding individuals defined as lethal by virtue of unsatisfactory DAD and StT scores, were minimal in all cases bar line JD generation F_2 which experienced proportionally high "mortality" due to StT scores. As a general observation, the exclusion of defined lethals can be expected to have negligible effect.

Statistical analysis of fitted response data showed no significant difference (at the 5% level) between inbreeder and outbreeder responses to selection, which is in agreement with qualitative assessment of observed responses. It should be noted, however, that the paired t-test of fitted responses gave a probability value of 0.0514 (i.e. a 5.14% chance that the difference between observed responses resulted from chance). While "statistically" a non-significant result, the probability is nevertheless sufficiently low to warrant further experimentation before confidently reporting equivalent responses to selection (for *B. carinata* and *B. juncea*).

Environmental considerations

Phenotype is the product of genotype and environment. This concept is commonly expressed as, $P = G + E$. The plant breeder effects selection upon the phenotype with the aim of improving the genotype (although it should be appreciated that isozyme selection methods are making it increasingly possible in some cases to select directly upon the genotype). Efficiency of phenotypic selection will be maximal if the environmental component of phenotypic variance is eliminated ($E = 0$). In practice, the environmental input to phenotype cannot be eliminated, and the best that can be achieved is to minimise environmental variation.

In order to make comparisons among members of the same generation, it is important that all members of that generation experienced common environmental conditions. This requirement was met by the experimental protocol. To make comparisons between successive generations, environmental conditions must be constant across generations. This condition was not met by the experimental protocol.

Genetic considerations

All *Brassica* spp. breeding lines used in this study are allotetraploid. Allotetraploid genomes are derived from the interspecific hybridisation of dissimilar diploid genomes. For morphological characters, such as those considered in this breeding experiment, it seems reasonable to expect allotetraploid Brassicas to behave as diploids, and response to selection should be analysed accordingly. This assumption would not necessarily be valid for the case of enzyme loci, where gene dose from dissimilar but related genes might have a quantitative effect.

It is assumed that plants in generation F_1, being derived from the crossing of either finished inbred lines or wild populations, will be fully heterozygous. In the case of the inbred lines, this assumption would be valid only if the inbred lines were mutually unrelated. Reckitt & Colman Ltd. declined to provide information about the genealogy of the breeding lines supplied, and in the absence of this information, the assumption that lines had no common pedigree cannot be substantiated.

Dispersive processes, a consequence of sampling and finite population size, become significant in small populations. In the experimental populations described, sampling variance and random drift, normally ignored for large populations, would be expected to have important effects on gene frequencies.

Fluctuation in population size among successive generations will reduce the effective population size still further. The effective population size, *N*e, can be calculated as follows:

$$1/Ne = 1/t[1/N1+1/N2+...+1/Nt]$$

where t is the number of generations, and N_1 to N_t denote the population sizes in the respective generations. It can be seen that the smallest populations have a disproportionately large effect on effective population size.

The experimental populations described were, by any standards, small. This was an inevitable consequence of limited greenhouse space, and hence unavoidable. Importantly, dispersive processes and inbreeding depression are inevitable consequences of the adopted method of selection. Phenotypic truncation selection, by reconstituting each successive generation from a small selected sub-set (breeding group) of the preceding generation, effectively reduces the size of the population to that of the selected group. Selection pressure was intense, and this severely reduces the effective population size. Taking the selection protocol used, $Ne = 2.18$. Dispersive processes can therefore be expected to be highly significant. The effect of dispersive processes on the population response to selection should not, therefore, be discounted.

For the outbreeding species, inbreeding depression is a likely consequence of reduced population size. A seemingly valid generalisation is that characters closely associated with the physical or reproductive fitness of the organism are most likely to deteriorate in response to inbreeding, whereas characters of little importance to fitness remain relatively unaffected. While the morphological characters under selection are likely to be relatively unaffected by the selection process, at a population level a reduction in viability and reproductive performance can be expected, as a consequence of the fixation of deleterious alleles. Fully inbred lines, observed to be both physiologically and reproductively viable, are descended from fit individuals within the population of overall declining fitness, having had advantageous or selectively neutral alleles fixed.

Mutation occurs in any population, and indeed constitutes the ultimate source of novel genetic variation upon which the plant breeder can operate. Unique, non-recurrent mutations, have little effect as an agent of evolutionary change. Recurrent mutations, an important source of genetic variation in natural populations, will occur at a characteristic frequency. In the absence of mutagenic agents, mutation rates are very small, typically in the region of 10^{-5} or 10^{-6} per generation (Falconer, 1981). In experimental-scale populations, mutation is unlikely to be detected. Mutation was not considered in this study. Migration forms another important source of genetic variation in natural populations. The genetic consequences of migration will depend on the rate of migration into the host population, and the genetic difference between the immigrants and the native population. In an experimental situation, migration can be excluded, as was the case for this programme.

General considerations

Artefacts of glasshouse cultivation were observed within the experimental populations. It was observed that root growth in *B. carinata* and *S. alba* was restricted by the 9" pots. In general, this root-binding occurred after the morphological characters under selection had been assessed, and therefore had no direct effect on the measured response to selection. The constriction of root growth can be expected to have an unpredictable effect on above ground morphology. There is no way of knowing whether the reproductive attributes of the plant were significantly affected.

All generations were prone to infestation by the Peach Potato Aphid, *Myzus persicae*. Various control methods were employed, dependent upon the severity of the infestation. The inconsistency among plant generations of the control measures adopted, clearly added an additional environmental variable to the experimental programme. The consistently most effective control measure was the use of a contact fumigant (pirimiphos-methyl smoke bombs) and a carbamate insecticide spray (ICI *Pirimor*). Malathion and nicotine sprays were generally ineffective. Aphid damage was characterised by wilting, presumably a consequence of sustained vascular damage, and in extreme cases the complete destruction of the plant reproductive structures. Due to the severity of the consequences of unchecked aphid infestation, breeding considerations had to be subordinated to the requirement of keeping the plants alive.

Additionally, infection of the breeding material by powdery mildew, genus *Erysiphe*, was commonly observed. There are several genera of powdery mildews. On cereals and grasses the genus *Erysiphe* predominates, whereas the genera *Sphaerotheca* and *Uncinula* are more common on fruit crops. Infections of Brassica crops are generally caused by *Erysiphe crucuferarum*, common on both wild and cultivated hosts. In nature, infection occurs in Spring and Summer - although in glasshouse populations infection would be expected to be independent of season - and is characterised by a white powdery growth on aerial plant structures. Despite high levels of infection, damage is largely cosmetic, and hence is only a significant problem in vegetable Brassicas. Bud deformation can result from extreme infections.

Unlike the cereals, where monogenic resistance has been used to control powdery mildew, resistance breeding in Brassicas has concentrated on classic polygenic (horizontal) resistance interactions between *E. crucuferarum* and its hosts. Resistance response rarely inhibits primary infection, but cell necrosis and deposition of lignin in the walls cause the

encapsulation of the haustoria, thus inhibiting fungal proliferation (L. Berry, personal communication). Mildew control was effected by using the systemic fungicide, Tilt Turbo (Ciba-Geigy), sprayed directly onto the plants approximately 2 weeks after germination, and again following pod set. The single exception to this was observed in experimental generation F_3, which suffered from a particularly severe mildew infection. Only partial control was achieved, and a high level of mortality recorded. Fortunately, the reproductive structures remained, in general, relatively unaffected.

In field characterisation of the parental material, the plants were seen to be highly susceptible to attack by Cabbage Root Fly, *Delia radicum*. This led to retarded growth, or death in most attacks. Root fly attacks resulting in retarded growth were taken to be lethal, and the plants rogued from the experimental plots. Observed mortality rates resulting from attack by *D. radicum* were typically between 10% and 17.5%. *B. juncea* variety J/1078/5/4/89/2013 was seen to be particularly susceptible, with mortality rates of 25% and 37.5% for the two field plots.

IV. SATSUMA (SPATIAL AND TEMPORAL SIMULATION UNDER MECHANISTIC ASSUMPTIONS): AN INTERACTIVE COMPUTER SIMULATION MODEL OF SPATIAL AND TEMPORAL PATTERNS OF DISEASE DEVELOPMENT IN CROP POPULATIONS.

INTRODUCTION

In order to better apply the principles and techniques learned through the development of PSELECT, modelling activities were transferred to a subject area where the underlying principles and mechanisms have been more comprehensively studied. Plant pathology, the study of the action and effect of pathogenic organisms on plant populations, has been important for as long as man has been cultivating plant species for food. Agents causing disease in plant populations, notably fungi, bacteria, viruses and invertebrates (nematodes, insects and mites), have two major effects on a host population: (1) by adversely affecting the health of the host plant, pathogenic agents are likely to reduce both the reproductive and vegetative performance of the plant. In the case of a cultivated edible plant species, this brings about a reduction in yield, an essential component in the "worth" of the plant for cultivation; (2) through variability in the genetically-determined susceptibility to pathogen damage, the causal agents of plant disease constitute an environmental pressure on the crop population, and are therefore instrumental, via differential selection, in causing change in the underlying genetic constitution of the host population.

Pathogen epidemics in crop populations have had significant medical and social effects on humans, which has contributed to the relative abundance of research in the field of plant pathology. As early as AD857, written records of ergotism (caused by the fungus *Claviceps purpurea*) claim the death of thousands of inhabitants of the Rhine valley (Carefoot & Sprott, 1967). Ergot of rye is mentioned repeatedly throughout historical records. As part of its life-cycle the fungus produces lysergic acid diethylamide (LSD), a hallucinogenic alkaloid which, if the infected grain is ground into flour for baking, can become rapidly distributed among a human population. Medical symptoms include abortion in pregnant females, fever, mental derangement and death. Numerous outbreaks of ergotism, sometimes named "Holy Fire" or "St. Anthony's Fire" have been recorded, with epidemics occurring as recently as 1951 in France (Fuller, 1968), where 32 cases of insanity and four deaths were recorded, and most recently a 1977 outbreak in famine-hit in Ethiopia (Demeke *et al.*, 1979). In the years 1845 and 1846, *Phytophthora infestans* (the causal agent of late blight of potato) resulted in the infection of much of the Irish potato crop, which subsequently rotted during Winter storage. The resulting famine caused approximately 1 million deaths through starvation, and resulted in the emigration of approximately 2 million to the New World (Woodhouse-Smith, 1962). Set against an initial population totalling approximately 8 million, the Irish Potato Famine can be seen as of one of the most significant crop epidemics of recent history. Late blight of potato was again to have dramatic effects on the affairs of humans, with a German epidemic in the years 1916-1917 causing severe food

shortages, the resulting demoralisation and civil unrest contributing to the end of the 1914-1918 war in Europe (Carefoot & Sprott, 1967). Between 1942 and 1943, leaf blight of rice, caused by the bacterial pathogen *Xanthomonas oryzae*, resulted in the Great Bengal Famine (Padmanabhan, 1973) and an estimated 2 million deaths through starvation.

While accounts of ergotism and potato blight have undoubtedly had among the most dramatic and significant effect on human activity in recent times, other crop epidemics are noteworthy through their financial effects. Taking a somewhat cynical point of view, it is arguable that these epidemics causing financial loss have contributed disproportionately to scientific advancements in the understanding and combat of crop epidemics. Between 1854 and 1860, powdery mildew of grape (*Uncinula necator*) caused great financial loss (Large, 1940), and through the importation of infected material from North America, the introduction of Phylloxera louse (*Daktulospharia vitifoliae*) to France, and the subsequent devastation of French vineyards (Lewin, 1993). Since this event, European vineyards have had to rely predominantly on grafting European variety scions onto North American, or North American hybrid root stocks resistant to the worst effects of Phylloxera. In the 1870's, coffee rust (*Hemileia vastatrix*) devastated the Ceylon coffee crop, which was cultivated predominantly as a genetically uniform plantation crop. By 1878 yields from Ceylon coffee plantations had fallen by 55%, the epidemic had spread to plantations across the Indian sub-continent, and the region's coffee industry was effectively destroyed (Carefoot & Sprott, 1967). This event was instrumental in promoting tea consumption in Britain, and in the establishment of coffee plantations in Central and South America. In 1970, 15% of the U.S. maize crop was lost to Southern maize leaf blight (*Bipolaris maydis*), a loss of approximately 20 million metric tonnes of corn with an estimated value of US$ 1 billion (Horsfall, 1972). Susceptibility to the pathogen was strongly correlated with hybrid lines having been produced utilising the *Tcms* male-sterility gene (Mercado & Lantican, 1961; Ullstrup, 1970), which caused alarm as hybrid lines of this type accounted for approximately 85% of total U.S. acreage.

Perhaps in consequence of the direct commercial and social effects of plant pathogens, the subject area has received a great deal of attention, and a great many of the mechanisms of plant diseases have been well characterised. Since the relationships between host plant, pathogen and the environment (which together define diseases of plants) can be quantified, the dynamics of plant disease epidemiology are amenable, theoretically, to computer simulation. Can a mechanistic computer model of plant disease therefore be developed from first principles, and in the event that it can, to what extent does its output correspond to observed patterns of disease spread in crop populations?

Principles of quantitative epidemiology

Temporal analysis

Van der Plank* (1963) is widely attributed with establishing plant disease epidemiology as a quantitative science. He showed that the logistic function could be used to describe epidemics caused by pathogens having multiple infectious generations per host growth cycle (polycyclic pathogens), and that the monomolecular function was suitable for the description of epidemics caused by pathogens having a single infectious cycle per host growth cycle (monocyclic pathogens). By introducing the logistic and monomolecular functions and carefully illustrating their applicability to the analysis of disease epidemics he provided plant disease epidemiology with a general scientific framework which could be used for theoretical development (Zadoks & Schein, 1988). Plant pathologists now had the means by which to compare the way different epidemics progressed over time and had parameters which could be used to make the comparisons quantitative and accessible to statistical analysis.

The logistic and monomolecular functions belong to a family of growth curves (Richards, 1959) which can be represented by the general formula:

$$y = \left[1 + \left(\left(1 - y_0^{1-m}\right) / y_0\right) \exp^{-rt}\right]^{1/1-m}$$

where y is disease at time t, y_0 is disease at time 0, r is the rate parameter, and m is a shape parameter. When $m = 0$ the function is equivalent to the monomolecular function, when $m = 2$ it is equivalent to the logistic function, and when $m = 1$ the Gompertz function (a special asymmetrical form of the logistic function) is obtained (Campbell & Madden, 1990). This family of curves is only one of the many which have been used to describe disease progress (Rouse, 1985), but all commonly used functions relate disease progress to three variables:

1. The amount of inoculum or disease at the beginning of observation (y_0, above).
2. The multiplication rate of the pathogen (included in r above)
3. The amount of uninfected plant material available for infection ($1-y_0$ above).

* From 1976, references to van der Plank are catalogued by the name Vanderplank.

The temporal analysis of epidemics has two main purposes. First, when used empirically, growth functions provide a basis for making predictions about the behaviour of diseases under known environmental conditions and thereby form the basis of practical forecasting systems. Secondly, where a particular growth function provides a good description of disease progress, the various parameters of the function can be given biological interpretations. In this way some insight can be gained into the biology of the pathogen. There are, however, dangers in placing biological interpretations on empirically derived functions, such as the general conception that any disease that can be described by the monomolecular function involves a monocyclic pathogen (Campbell & Madden, 1990). An empirically fitted function, while accurately describing the temporal development of an epidemic, gives no information regarding the reproductive behaviour of the pathogen. For example, *Phytophthera fragariae* (the causal agent of raspberry root rot) is a polycyclic pathogen whose annual epidemics can be described by the monomolecular function.

Spatial pattern analysis

Van der Plank's contribution to the temporal analysis of epidemics is apparent in current epidemiological methodology. In addition to his description of epidemics with respect to time, he highlighted the importance of the spatial aspects of epidemics in his original treatise (van der Plank, 1963) and in subsequent work (van der Plank, 1975). Despite this early recognition of the spatial components of epidemics, the development of methods for analysing the spatial spread of diseases has lagged behind developments in temporal analysis.

The way in which a pathogen is spatially distributed in a crop may affect the rate at which it causes new infections, and the appearance of new infections will of course have a feedback effect on the spatial pattern of the disease. In this way the temporal and spatial components of disease are mutually dependent. The spatial pattern of a disease has practical importance for several reasons. First, the pattern of diseased plants in a field affects the accuracy of sampling protocols on which disease progress data are based; and so it can affect threshold-based disease management decisions. Secondly, the degree of patchiness (aggregation) of a disease can influence its effect on yield (Blodgett, 1941) as a result of compensatory growth by the healthy neighbours of diseased plants.

After achieving initial popularity in the 1930's and 40's (Cochran, 1936; Blodgett, 1941) interest in the spatial aspects of epidemiology evaporated, only to resurface very recently, despite the attention drawn to them by van der Plank (1963; 1975). While temporal analysis

methodology is well established and fairly uniform in application, spatial pattern analysis is an actively evolving field of endeavour, and methods for spatial analysis are more diverse than those for temporal analysis. In any situation the most suitable method will depend on the aims of the researcher (Campbell & Noe, 1985).

Techniques based on mapping can be extremely powerful and allow both detailed empirical analysis and determination of underlying biological processes. An initial step with mapped disease data could consist of the imposition of a colour or shading scale, to quadratised disease incidence or severity data (Campbell & Madden, 1990; Campbell & van der Gaag, 1993; Ristaino *et al.*, 1993). A further stage of quantification with quadrat data is to examine the fit of disease severity or incidence data to various probability distributions. The aim with this approach is often to assess whether disease is aggregated or occurs randomly in the sampled area. In the case of count data (for example, lesions per leaf) a random distribution is indicated by a good fit to the Poisson distribution, while aggregation is indicated by a poor fit to the Poisson distribution but a good fit to the negative binomial distribution (Strandberg, 1973). Where disease is assessed for presence/absence, the data are commonly referred to as "incidence" data (Nutter *et al.*, 1991). The analogous distributions in the case of incidence data are the binomial (random distribution) and the beta-binomial (aggregated distribution) (Hughes & Madden, 1992; 1993; Madden & Hughes, 1994).

While frequency distribution data methods can be used to characterise the degree of aggregation of a disease they cannot generally give any indication of the biological processes which have resulted in the measured distribution (Campbell & Noe, 1985). Further quantitative methods which can be used to describe spatial aspects of epidemics include lagged correlation analysis (Campbell & Noe, 1985; Gottwald *et al.* 1992) and distance class analysis (Nelson *et al.* 1992). Lagged correlation analysis can provide information about the size and orientation of clusters of disease based on disease severity data while 2-dimensional distance class analysis gives similar information from disease incidence data. In both cases the results can be useful in characterising the way a disease spreads, and may reveal or suggest features of the biology of the pathogen (Campbell & Noe, 1985).

A more recently adopted method which has some similarity to lagged correlation analysis is the use of geostatistics. The basic method here is to calculate the variance in disease severity or incidence at various distances from selected diseased plants. By analysing the

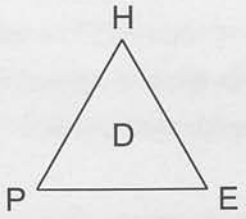
variance/distance relationship in this way it is possible to determine the distance over which disease is spatially dependent (Chellemi *et al.*, 1988; Kocks & Ruissen, 1994).

Despite the variety of analytical methods available for spatial and temporal analysis of epidemics the construction of combined spatio-temporal models has been relatively slow. Jeger (1983; 1986; Jeger *et al.*, 1983) and subsequently van den Bosch *et al.* (1988) have developed models for the description of expanding disease foci in time and space, but not all diseases conform to this description. Waggoner & Rich (1981) suggested a more general approach, suitable for disease severity assessments, in which the effect of disease aggregation on the rate of disease increase was accounted for by combining the negative binomial distribution with the logistic function. A similar approach has been suggested for incidence data (McRoberts & Hughes, 1994). Despite these theoretical developments spatio-temporal models are not routinely applied to the analysis of disease epidemics and the standard approach is to perform separate temporal and spatial analyses (Campbell & van der Gaag, 1993; Ristaino *et al.*, 1993).

To summarise briefly, the principal motivation of plant disease epidemiology is the succinct description of the spread (spatial) and progress (temporal) of plant diseases for the purposes of improved understanding of the underlying biology and disease management. Traditionally both the temporal and spatial components of disease epidemics have been analysed empirically.

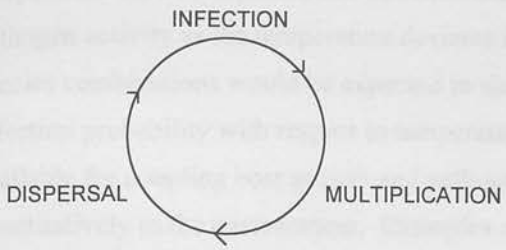
Fundamental processes

Plant diseases are the result of the interaction between a pathogen (P), a host plant (H) and a suitable environment (E). This concept has been described in the plant disease triangle (Wheeler, 1976).



Any change in disease in time and space is an epidemic; not all epidemics are short-term dramatic increases in disease.

With respect to the pathogen, three processes are involved in the development of a disease: (1) infection; (2) multiplication; and (3) dispersal. In a very simplified representation these processes can be seen as a cyclical chain of discrete events:



The overall rate at which the pathogen population increases (which will be directly related to the rate at which disease is seen to increase) will be determined by the rates of these three processes.

Numerous features of the plant will contribute to the rate of a disease epidemic (Rouse, 1988; Hau, 1990). The effectiveness of genetically-determined resistance may affect both the rates of infection and multiplication of the pathogen. Growth of the host plant will affect the rate of the disease epidemic in several important ways. Crop growth may affect disease resistance; the phenomenon of different levels of resistance in juvenile and adult tissue is well known (Simmonds, 1979; Hau, 1990). Secondly, the rate of the epidemic is related to the amount of tissue available for infection, so continued growth by the crop will affect the rate of disease progress by providing fresh tissue for infection. In practical terms, the final yield of the plant will be determined by the balance between the crop's ability to produce new tissue, the detrimental effects of disease, the rate of disease increase, and the nature of the relationship between the level of disease and the degree of physiological disfunction it causes. Rouse (1988) has pointed out that models of epidemics which are intended to provide quantitative analyses of the effect of disease on yield are more accurate if they are based on crop growth rather than on descriptions of disease symptoms. Despite the importance of processes involving the host plant to disease epidemics, host growth is still largely ignored in many epidemiological studies, perhaps as a result of the complexities involved in coupling host and pathogen growth (Rouse, 1988; Hau, 1990).

The host and pathogen processes involved in disease epidemics are influenced by their physical environment. This influence is often stated qualitatively with reference to the disease triangle. The qualitative perspective (that disease only occurs in a suitable environment) is of course a generalisation and in reality the processes which drive an epidemic are quantitatively related to the environment, principally to temperature and to a

lesser extent humidity (Hau *et al.*, 1985, Campbell & Madden, 1990). The influence of temperature on epidemics is directly attributable to its effects on the biochemical processes of metabolism and growth, with a given pathogen having a characteristic optimum temperature for development and infection activity, typically with a non-linear reduction in pathogen activity as the temperature deviates from the optimum. Different pathogen-host species combinations would be expected to show characteristically different curves for infection probability with respect to temperature. Hau (1990) has discussed the methods available for coupling host growth and pathogen growth to each other and relating both quantitatively to the environment. Examples of practical analytical models which combine host (though not necessarily growth), pathogen and environment are relatively common and have been reviewed by Coakley (1988).

It has already been noted that the design and form of any model will be determined by the aims of the modeller. The essentially analytical empirical models which have been discussed so far have generally been motivated by a desire to describe and predict disease development and its effect on crop yield. While simulation models can also achieve these objectives the motivation for their development may be rather different (Teng, 1985).

History of plant disease simulation

Unlike the situation for the development of selection models, examples of computer simulation models in plant pathology are relatively commonplace. Among the earliest published examples are a range of computer-based simulation models developed by Paul Waggoner and co-workers in the late 1960's and early 1970's. EPIDEM (Waggoner & Horsfall, 1969) was arguably the first simulation exercise to receive widespread acceptance among academic plant pathologists. The model simulated the action of early blight of tomatoes (*Alternaria solani*). This model was preceded by EPDEM, similar to EPIDEM in most respects, except that the target of simulation was that perennial favourite of mycologists, late blight of potato (*Phytophthora infestans*). The last model from the initial phase of this prodigious programme was EPIMAY (Waggoner *et al.*, 1972; Shaner *et al.*, 1972), simulating the action of corn leaf blight (*Bipolaris maydis*).

Although certainly pioneers in this field of research, Waggoner and co-workers were not alone in the formative era of plant disease modelling. Important early contributions to the establishment of computer-based simulation modelling as a valid contribution to the science were made by Zadoks, Rijsdijk and Shrum. EPISIM simulated the action effect of yellow rust (*Puccinia striiformis*) in wheat crops, and was in various forms described jointly and

separately by Zadoks and Rijsdijk (Zadoks, 1971; Zadoks & Rijsdijk, 1972; 1973; Rijsdijk, 1975).

The earliest plant disease simulators were strictly deterministic, with the algorithm having been empirically derived from experimentally-derived data, and hence pathogen-host specific. The first simulation model purporting to be pathogen-inspecific was developed by Shrum (Shrum, 1975) at Pennsylvania State University. The model, unsportingly named EPIDEMIC, was developed in FORTRAN and intended for use on the large mainframe computers becoming available in the 1970's. EPIDEMIC was vastly superior to its predecessors in a number of ways. Most significantly, EPIDEMIC was designed to be system-inspecific. Prior to this, a given observational data set was examined, and a computer model developed to conform to these data. Models derived in this manner are strictly empirical and more often than not, deterministic. In this case, if modelling objectives or opinions were to change, or if new or previously discarded information were to be considered as important, a major reprogramming effort would be required to update the computer model. EPIDEMIC was an early example of a modular simulator. Important events or processes in the development of a crop disease epidemic were coded in separate modules. The simulator could then achieve a degree of disease-inspecificity by controlling which of the modules pertained in a given simulation. In terms of structure, submodels of EPIDEMIC were arranged hierarchically. Information produced by one submodel is fed as a single item to the subsequent submodel where it forms a component input variable for the calculation of the next item. A degree of directional independence is maintained, with each subsequent module within the simulator remaining unaffected by the complexity or otherwise of the preceding calculations.

EPIDEMIC was also a significant milestone in the evolution of plant disease simulation models in that it was the first model to introduce a spatial treatment of disease development, in addition to the then more common temporal description of disease progress.

"An epidemic is a spatial as well as a temporal phenomenon. The assumption that there is uniform distribution of disease in a field is seldom true, and simulators based on such an assumption may eventually have to be modified to achieve the sensitivity needed for adequate assessment of many diseases. By definition, adjacent points of a contagious disease are not independent... Since the degree of influence exerted on (and by) adjacent points varies with time and space, this dependence is potentially as important a variable as any other, and should not be ignored." (Shrum, 1975).

Despite being very dated (input variables were specified using an ordered stack of punch-cards) and now technologically surpassed, EPIDEMIC marked a crucial stage in modelling

methodology. Sadly, this excellent model was developed as a doctoral research programme, and appears to have undergone no subsequent development.

Perhaps in consequence of the relatively high value of the crops, and their perennial habit and consequently time-consuming and expensive experimental attributes, fruit crops have received a great deal of attention from computer simulation modellers. The mark was established by Kranz and his co-workers, with the model EPIVEN (Kranz *et al.*, 1973) simulating apple scab (*Venturia inaequalis*). Later developments in simulating this disease were presented by Arneson *et al.* (1979), and Kranz (1979). Grapes (*Vitis* spp.) have also come in for significant attention, with the model MELSIM (Ferris 1976; 1978) addressing the problem of grape root knot (*Meloidogyne* spp.), and the excellent but unnamed work of Sall (1980a; 1980b) on the viticulturist's eternal nightmare, grape powdery mildew (*Uncinula necator*). Sall's work, along with that of Teng and co-workers (Teng *et al.*, 1980) modelling barley leaf rust (*Puccinia recondita*), was significant in introducing a stochastic element to model output. All models to that point, whether by design or by omission, had been deterministic.

Other named computer-based simulation models worthy of mention include EPICORN (Massie, 1973) a simulator of Southern corn leaf blight; CERCOS (Berger, 1976) modelling Cercospora blight (*Cercospora apii*) in celery; EPISEPT (Rapilly & Jolivet, 1976; Rapilly 1977; 1979), a simulator for leaf and glume blotch of wheat (*Septoria nodorum*); EPIMUL (Kampmeijer & Zadoks, 1977), simulating epidemics in mixed host populations; EPIPHTORA (Gurevich *et al.*, 1979), SIMPHYT (Stephan & Gutsache, 1980) and LATEBLIGHT (Bruhn *et al.*, 1980; Bruhn & Fry, 1981), all simulating late blight of potato; EPIGRAM (Aust *et al.*, 1983) and GEMETA (Hau, 1985), both modelling barley powdery mildew (*Erysiphe graminis*).

Software for teaching: simulation vs. analytical schools

Hau (1990) has illustrated that at a complex level of development, models which are essentially analytical in purpose may be similar to simulation models in outcome, if not in developmental philosophy. It is perhaps understandable, given the research orientation of the majority of plant disease epidemic models, that there is little literature discussing the ideal specifications of computer software for teaching the principles of epidemiology. However, the choice of approach can only be between an analytical model, a simulation model, or a combination of the two.

Hau (1990) discussed the divide between the simulation and analytical schools in plant disease epidemiology. The analysts take their approach from van der Plank's (1963) original contribution. The underlying philosophy of this school is that systems of simple parameter-sparse equations can give accurate descriptions of plant disease epidemics. One of the benefits of this approach is that the small number of parameters used to describe the epidemic can be very accurately estimated from empirical observation (Hau, 1990). Taking as an example the logistic function introduced by van der Plank (1963):

$$dy/dt = r \cdot y \cdot (1-y)$$

all of the environmental and host variables which determine the rate of the epidemic are summarised by the rate parameter, r. Van der Plank drew attention to this:

"*Every contribution, whatever its cause, is pooled in one single comprehensive figure that estimates the rate, and in the pool it loses its identity.*" (van der Plank, 1963).

For the epidemiology teacher the analytical approach offers mathematical simplicity. It is, however, vital to establish in the minds of the students the importance of each of the interacting processes which drive an epidemic, and this is hindered if the model used for the teaching is one in which they have lost their identity.

The contrast between the analytic and simulation schools can be seen in Teng's (1985) description of the term, "simulate":

"*To simulate means to duplicate the essence of a system or activity without actually attaining reality itself.*" (Teng, 1985).

Rather than fundamental processes being subsumed into one or many parameters they are described by a series of (usually complicated) model sub-components each of which may contain deterministic or stochastic elements. While this approach will certainly ensure that students must consider the processes and their interactions care must be taken that the complexity of the model does not obscure the biology on which it is based. The danger is that the processes, safe from being lost in a characterless pool, are washed away in a torrent of formulae, model jargon and elaborate prose.

OBJECTIVES

An appreciation of epidemiological principles is a pre-requisite to a comprehensive understanding of the interaction between pathogen and crop populations. A potentially useful approach is to introduce the subject, without specific reference to mathematics, through a computer model which simulates the spread of a fungal pathogen (McRoberts & Partner, 1994). Once a graphical introduction has been concluded, the mathematical description of disease progress becomes more readily understandable by reference to the previously observed spatial and temporal patterns. It may be the case that the ideal teaching software should have characteristics of both the analytical and simulation approaches. The following characteristics should be apparent in suitable software:

1. At an introductory level the program should allow a non-mathematical consideration of epidemics.

2. The key variables/processes which influence the rate of epidemics should be explicitly included in the software to force the user to consider them.

3. While the fundamental processes should be considered in the software the user should be protected from the mechanics of the model routines to which they correspond.

4. The data generated by the model should be appropriate for illustration of the methods used in the quantitative analysis of epidemics.

5. The software should be suitable for use at increasing levels of complexity as users gain experience of the subject area.

6. The software should be robust, have a relatively simple interface and be undemanding in its hardware requirements.

There are a number of computer programs for the PC which deal either with plant disease epidemics or the population dynamics of epidemics in a general way. Some of these have been devised specifically with educational application in mind, for example LATEBLIGHT (Bruhn *et al.*, 1980; Bruhn & Fry, 1981) or POPULUS (Alstad *et al.*, 1991), but these either deal with specific pathogen-host systems or are not specific to plant disease epidemiology. Importantly, no currently available simulator of plant disease epidemics offers anything but

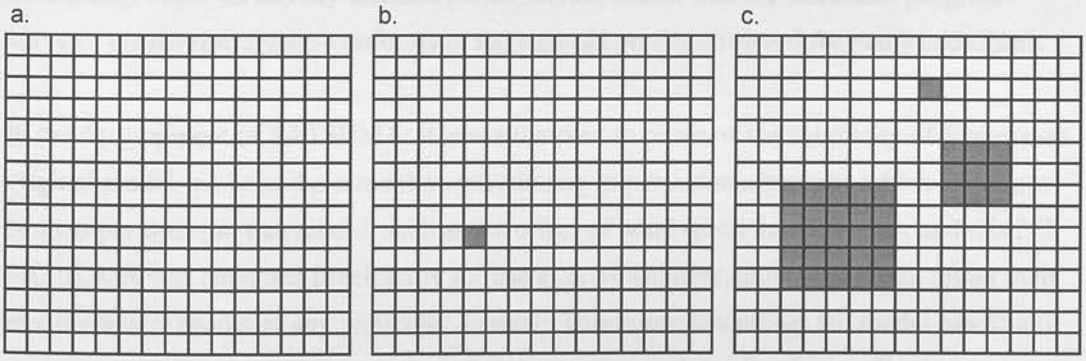
a temporal treatment of epidemics, and the requirement for a spatial simulation model is pressing.

The program presented here can be used as a teaching aid at several academic levels. At an introductory level users can be asked to consider the effects of environmental variables on the rate and pattern of disease spread, and then test their assumptions by using the model to give a qualitative description of a given epidemic. This can then form the starting point for more detailed discussions of a quantitative nature. A quantitative element can be introduced by virtue of the numerical output that accompanies the pictorial evolution of the epidemic. By running a series of simulations, systematically altering one of the environmental variables (e.g. temperature), a simple sensitivity analysis can be conducted to asses the effect of that variable in the development of an epidemic. More advanced examination of the mathematical basis of epidemiology can be conducted with reference to the model's numerical output, allowing practice in analytical methods for the spatial and temporal description of epidemics and consideration of the difficulties in combining these types of analysis to produce spatio-temporal models; this last subject being the current distraction of many plant disease epidemiologists.

SIMULATION

The computer simulation model, SATSUMA (Spatial And Temporal Simulation Under Mechanistic Assumptions), has been developed for use in the examination of basic epidemiological principles, with specific reference to the spread of fungal pathogens in crop populations. The model operates on a series of user-defined parameters, and through graphical representation via the computer screen, the spatial and temporal patterns of pathogen spread through the crop population can be observed. The three major classes of fungal pathogen, air-borne, splash-dispersed and soil-borne, can be simulated. Disease progress data are fed to an output file for presentation in tabular format. Underlying principles, assumptions, approximations and potential improvements to the model are fully discussed.

The model is a direct descendent of the "pen and paper" teaching game devised by Professor N.W. Simmonds, formerly of the Edinburgh School of Agriculture. In this game a grid of boxes (Figure 4.1a) was used to represent a field or group of plants. One box was allocated as the primary infection focus, and shaded in (Figure 4.1b). All adjacent boxes were then shaded to represent the first cycle of infection, and the epidemic progressed until all plants became infected. Secondary infection foci could be initiated by reference to random number tables, and the disease spread from these secondary foci in an identical manner to that described for the primary infection focus (Figure 4.1c). The students gained an appreciation of spatial and temporal aspects of disease development before being required to consider the mathematical description of the epidemic.



Figures 4.1a-4.1c. Description of Simmonds' teaching model, showing the simulated field (a), the primary infection focus (b), and the ensuing epidemic after two full infection cycles (c), with a single secondary infection during each cycle.

The aim in the development of SATSUMA was to preserve the simple interactive style of Simmonds' analogue model, but to improve the model by the incorporation of additional features not able to be included in the original model.

Key features of Simmonds' teaching model:

1. Clear visual display of spatial development of epidemic
2. Intrinsically sigmoidal temporal response
3. Neighbour-to-neighbour primary infection mechanism
4. Random generation of secondary infection foci

Circumvention of limitations of the original model through computerisation:

1. Simulation of important environmental variables: temperature, wind speed and direction, rainfall
2. Stochastic element by virtue of probability components
3. Targeted secondary infection foci
4. Larger crop population can be simulated
5. Automated data output

The algorithm contains no formal statement of the relationship between time and disease incidence. The physical structure of the model, and the basic idea that each infectious plant has the potential to infect all its neighbours, coupled with the loss of secondary infections should they occur on already infected plants, should ensure that the simulated progress curve of the disease always conforms to the sigmoid profile observed for many pathogens.

In the development of SATSUMA, it was attempted to retain of the simplicity of Simmonds' original model, while at the same time introducing important mechanisms which could not be incorporated into that model, such as the effect of wind speed and direction, and rainfall. SATSUMA was designed specifically for use as an illustrative educational tool, rather than as a predictive model of any particular disease. This notwithstanding, the model has found research application, and it is anticipated that more research applications will arise.

System

SATSUMA was written in Turbo Pascal version 6.0 (Borland International Inc.), and was coded and compiled on an IBM PS/2 55sx personal computer (IBM Corporation), equipped with 4 MBytes Random Access Memory (RAM). No math co-processor was fitted. Designed for IBM and compatible personal computers, the compiled program runs in the DOS environment (Microsoft Corporation).

The program was designed to be suitable for use on relatively low-specification personal computers. The model was tested on a wide range of machines, and while more powerful microprocessors were seen to improve performance (particularly with respect to the speed at which the graphical representation of the field was refreshed after each infection cycle) there are no unusual hardware requirements. Minimum system requirements are DOS 3.3 or higher, 512kBytes RAM, EGA graphics adapter or better. The model runs from floppy disk if required to do so, but with the anticipated reduction in performance, due primarily to system overheads associated with disk access.

Algorithm

SATSUMA operates upon user-defined parameters to generate a symbolic crop population, or field. The field is represented graphically on the computer screen, with the four states of plant condition, HEALTHY, INFECTED, INFECTIOUS (a sub-set of infected) and DEAD, being represented by mutually distinguishable characters within the simulated field.

The user-interface consists of a series of simple data input screens, prompting the user for the parameter values from which the simulated epidemic will be generated. The simulation acts on the following user-specified variables:

Integer variables:

1. Field dimensions, X by Y, giving the total number of plants in the field
2. X-Y co-ordinates of the primary infection focus
3. In the event of air-borne or splash-mediated pathogen dispersal: the variation in wind direction, given as an arc about the prevailing direction
4. A "noise" variable, to mimic unspecified environmental effects

Real variables:

1. Mean temperature and its standard deviation across the pathogen life-cycle
2. For the case of wind-mediated pathogen dispersal: mean wind speed and its standard deviation
3. For the case of wind-mediated pathogen dispersal: threshold wind velocity for spore release

String and character variables:

1. Mode of pathogen dispersal
2. For air-borne or splash dispersal: prevailing wind direction
3. For splash dispersal only: rainfall

All other variable values used in calculations within the simulation, are either generated or calculated internally within the model. Calls to the compiler RANDOM function yield uniform distribution random integers. Where normally distributed values are used, transformation of system-supplied values is by the Marsaglia-Bray Polar method (Marsaglia & Bray, 1964), as this method was shown to give satisfactory transformation of uniform-distribution deviates supplied by the Turbo Pascal compiler (Partner *et al.*, 1993). This notwithstanding, Box-Muller transformation (Box & Muller, 1958) and transformation by the rules of the Central Limit Theorem, or any other suitable method of generating normally-distributed deviates could be used if so desired.

The following environmental variables are generally considered the most important in simulation models of pathogens:

1. Temperature
2. Humidity
3. Wind speed
4. Wind direction

In biological terms, the importance of these variables with respect to the rate of pathogen spread is: Temperature > Humidity > Wind speed (Campbell & Madden, 1990). Wind direction is important only to the pattern of the epidemic, and in isolation, does not affect its rate. Wind direction will influence the rate of an epidemic in consequence of coincidental

factors, such as proximity of the infection focus to the edge of the field, or to previously affected plants.

Temperature

Temperature is the dominant driving variable in the model. Temperature-dependent plant disease simulation models and pathogen growth models are common. (Hau *at al*, 1985; Teng, 1985; Campbell & Madden, 1990). A wide range of functions have been examined for relating temperature to pathogen development. Generally the influence of temperature is accounted for by including a temperature response curve in the model for host and pathogen (Hau *et al.*, 1985) although more complex approaches have been examined (Waggoner & Parlane, 1974a; 1974b). These functions typically relate temperature to actual measurements of pathogen growth; few introduce a probabilistic element (Hau *et al.*, 1985). Among the simpler models for the effect of temperature on pathogen development are quadratic functions which produce symmetrical parabolas (Shearer & Zadoks, 1974; Chellemi & Marois, 1991).

In SATSUMA, an assumption is made that the probability of infection is directly proportional to the growth rate of the pathogen. An idealised temperature function is used. A second order polynomial, with parameter values selected to define a range of probabilities of pathogen growth between 0 and 100% for temperatures between 0 and 35°C, is used to define pathogen activity at each cycle, with all other variables acting after the underlying activity of the pathogen has been established by the temperature.

The quadratic function:

$$P_g = 3.5 + 10(T) - 0.26(T^2)$$

where P_g = probability of pathogen growth, and T = temperature (°C), gives a symmetrical parabola which reaches its apex around 19°C (Figure 4.2). This is defensible on theoretical grounds: it exhibits all the important features of a pathogen temperature-activity profile; i.e. it goes up then down, with activity rising to a maximum at an optimal temperature before decreasing at higher temperatures. We can expect the relationship between temperature and pathogen activity to be pathogen-specific, and it might be an interesting exercise to see how different temperature functions affect the model output.

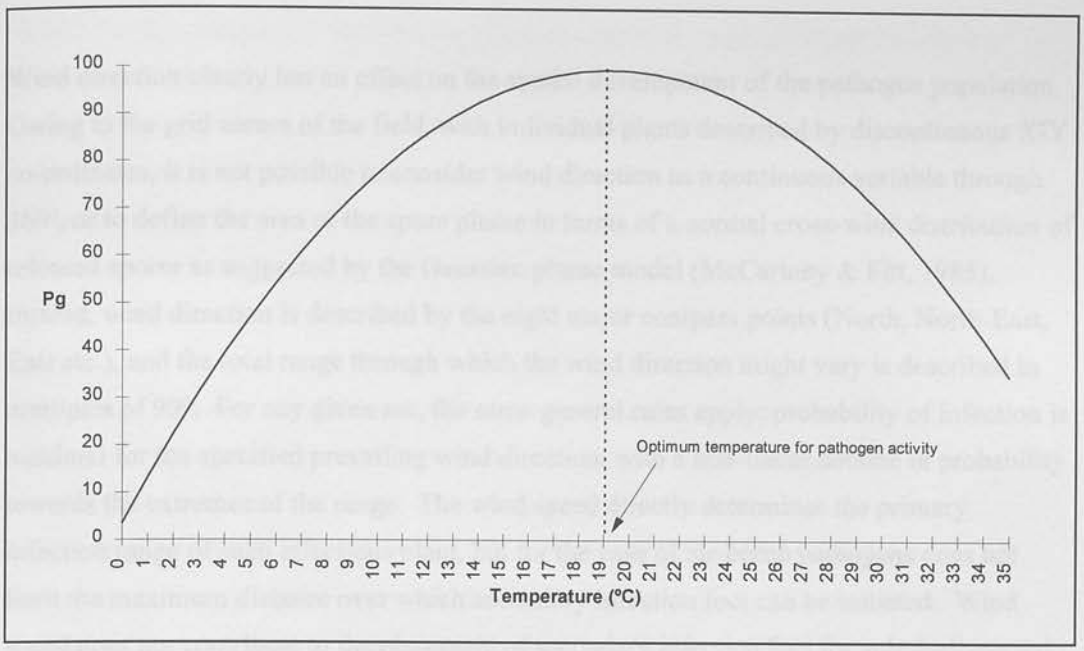


Figure 4.2. Graphical representation (0-35°C) of the relationship between temperature and probability of pathogen growth (underlying activity).

Wind speed and direction

The importance of wind speed and direction in determining the spatial pattern of disease spread from an infection focus was recently discussed by Zawolek (1993). Spore dispersal involves three processes: release, transport and deposition (McCartney & Fitt, 1985). The faster the wind blows, the further liberated spores can travel. As an additional consideration, at higher wind speeds wind-borne spores will have higher velocities, and consequently a higher efficiency of impact on the target host (Gregory, 1961). These effects of wind speed operate in conjunction with each other, with the rate of epidemic being determined by the number of spores liberated, their distance travelled, and the efficiency of infection upon coming into contact with a potential host plant. In the model, wind speed affects spore dispersal through the processes of release and transport. The effects of wind speed on spore deposition are not considered. Some fungi have a threshold wind speed below which there is no significant release of spores, while others discharge spores continuously (Aylor *et al.*, 1981). Both mechanisms can be simulated in the model.

The mean wind speed and its standard deviation are specified, and a realised wind speed calculated each pathogen development cycle in a similar manner to that described for temperature. If a minimum wind velocity (threshold velocity) is specified, realised wind speeds less than this threshold will preclude infections for a complete infection cycle.

Wind direction clearly has an effect on the spatial development of the pathogen population. Owing to the grid nature of the field, with individual plants described by discontinuous X-Y co-ordinates, it is not possible to consider wind direction as a continuous variable through 360°, or to define the area of the spore plume in terms of a normal cross-wind distribution of released spores as suggested by the Gaussian plume model (McCartney & Fitt, 1985). Instead, wind direction is described by the eight major compass points (North, North-East, East etc.), and the total range through which the wind direction might vary is described in multiples of 90°. For any given arc, the same general rules apply: probability of infection is maximal for the specified prevailing wind direction, with a non-linear decline in probability towards the extremes of the range. The wind speed directly determines the primary infection range of each infectious plant, but for the case of air-borne pathogens does not limit the maximum distance over which secondary infection foci can be initiated. Wind speed does not contribute to the placement of secondary infection foci for splash-dispersed pathogens.

Rainfall

Rainfall, or more specifically humidity and rainfall, will clearly be of importance in the case of water mediated spore dispersal. Moisture affects fungal proliferation in two major ways: firstly, if there is insufficient moisture the growth and reproduction of the pathogen may be inhibited, and its infectious potential therefore greatly reduced; secondly, in the case of splash dispersal of spores, rainfall is needed to provide the medium by which the spores are brought into contact with target plants. It is reasonable to assume that maximal spore dispersal is achieved at an optimum level of rainfall, the relationship being described in a similar way to that for rate of infection and temperature.

The Gaussian plume model for air-borne dispersal is not applicable for the dispersal of splash droplets. Although wind and rain are known to interact in the dispersal of splash-dispersed pathogens, the significance of the wind within the crop canopy is unclear (Fitt *et al.*, 1989). Field observations on a number of pathogens have shown that, despite the mainly ballistic nature of the spread of spore-carrying water droplets, wind direction does have an influence on the dispersal of splash-dispersed pathogens (Fitt *et al.*, 1989). Although the effects of wind speed are unclear in relation to splash-dispersal, there are observations that indicate that splash-dispersed spores are carried further in moving air than in still air.

In SATSUMA, correlation between the effective dispersal range of the spores and the amount of rainfall during the pathogen life-cycle is assumed. Three potential infection ranges are specified in response to the level of rainfall. Three alternative levels of rainfall over the season, DRY, TYPICAL, or WET, can be specified by the user, with the potential range of infection increasing with the level of rainfall.

Pathogen dispersal

Given mean and standard deviation values for temperature and wind speed, along with a predominant wind direction and an arc through which the wind direction will vary, and in the case of splash-dispersal an indication of the rainfall, every complete pathogen development cycle will be subjected to set of environmental conditions which will determine the pattern of the epidemic. SATSUMA simulates the behaviour of the three major modes of dispersal of fungal pathogens of crops: (1) air-borne (wind dispersal), (2) water-borne (splash dispersal) and (3) soil-borne.

(1) Air-borne pathogens

Air-borne pathogens have received the greatest attention from simulation modellers in plant pathology. The explosive nature of epidemics caused by air-borne pathogens results from their very high rate of spore production on a suitable host plant, coupled with the high potential for widespread dispersal which can occur under favourable weather conditions. The most studied genera include *Phytophthora* (Bruhn & Fry, 1981), *Erysiphe* (Hau, 1988), and *Puccinia* (Zadoks & Rijsdijk, 1972; Shrum, 1975).

The rate of disease increase is determined by temperature and wind speed. An infectious plant has the ability to infect any and all of its immediately adjacent neighbours, and those non-adjacent neighbours situated downwind within an arc centred about the realised wind direction. The maximum distance at which neighbouring plants might be infected is determined proportionally by the realised wind speed. The infection of immediately adjacent neighbours is independent of wind speed and direction.

(2) Splash-dispersed pathogens

Simulation models of splash-dispersed pathogens are less common than those of air-borne pathogens, but there are a few well-known examples, notably the simulations of *Venturia* (Kranz *et al.*, 1973) and *Septoria* (Rapilly, 1977). In SATSUMA, the rate and pattern of

pathogen dispersal in the model is determined by the temperature, rainfall, and wind direction. With each infection cycle, an infectious plant has the ability to infect any and all of its immediately adjacent neighbours, and those non-adjacent neighbours situated downwind within an arc centred about the realised wind direction. The maximum distance at which neighbouring plants might be infected is determined proportionally by the degree of rainfall. The infection of immediately adjacent neighbours is independent of rainfall and wind direction.

(3) Soil-borne pathogens

Campbell and Madden (1990) pointed out the common misconception that all soil-borne pathogens are monocyclic. Gilligan (1985) noted that most soil-borne pathogens are either oligocyclic or polycyclic. The most commonly used mathematical models for describing the development of soil-borne pathogens are therefore the same as those for air-borne pathogens.

In SATSUMA, explicit consideration of the pathogen's reproductive behaviour is purposely avoided. It is stressed that the infection process for soil-borne pathogens occurs predominantly from neighbour to neighbour, with new infection foci arising relatively infrequently. The rate of pathogen dispersal is determined by the temperature and the level of crop susceptibility to the pathogen. With each infection cycle an infectious plant has the ability to infect any and all of its immediately adjacent neighbours. An homogeneous soil structure is assumed, and hence an equal potential for disease spread laterally and longitudinally. Infection is independent of rainfall and wind variables.

(4) Secondary infections - air-borne and splash-dispersed pathogens

Secondary infections may arise in consequence of favourable environmental conditions (the conditions being determined by the environmental variables relevant to the mode of dispersal). The rate and pattern of disease spread from these secondary infection foci are subject to the same rules as described for the primary infection focus.

Subject to the constraints imposed by the environmental conditions and the mode of pathogen dispersal, SATSUMA places secondary infections randomly within the field, and determines which are allowed to proceed by determining whether the target plant is downwind from the infectious plant currently under consideration. While this does not strictly model the mechanism of spore dispersal from an infectious source to potential

targets, it is logically comparable and has the not insignificant advantage of being computationally simple. The mechanism used results in a sufficiently realistic pattern of spread for use in an educational program.

Progression of infection

The four states of plant condition, HEALTHY, INFECTED, INFECTIOUS and DEAD, are chronologically related in the order shown. The simulated pathogen life-cycle comprises two distinct phases: a maturation phase and an infection (reproductive) phase.

In the maturation phase, an infected host plant has the potential to become infectious, and previously infectious plants can die. It is assumed that plants will only degenerate to the next condition in the sequence. For example, in consequence of the pathogen maturation phase an infected plant might become infectious, but would not be able to degenerate to death without first having passed through an infectious stage (and hence taking a minimum time of two complete pathogen life cycles to do so). Progression through conditions is unidirectional, so that a plant, once infected, has no chance of recovery to the healthy condition. By extension, an infected plant will ultimately die, the time taken being determined by environmental conditions and random factors. The probability of degeneration from one condition to the next is determined by the variable parameters listed above, and involves a random component.

In the infection phase, infectious plants have the opportunity to infect their neighbours and to initiate secondary infection foci at more distant points in the field. Healthy target plants can, subject to favourable environmental conditions, become infected. Existing infected, infectious, or dead plants will be unaffected by the pathogen's reproduction and dispersal activities. The assumption is made that dead plant material is no longer infectious. The model is closed, with no infections from external sources being permitted. In addition to the graphical summary presented, disease progress data are fed to an output file, where they can be used as the starting point for a mathematical description of the epidemic.

Design & development

A dictionary of variable names used in SATSUMA is given as Appendix D1. The program code for SATSUMA, listed as Appendix D2, follows a common Pascal structure similar to that described for PSELECT. Executable component routines were coded as discrete procedures, in addition to which, a range of custom-made functions was specified to

enhance the interactive potential of the model. The main program block calls the coded procedures, the executable order of which determines the logical progression through the model. A diagrammatic representation of SATSUMA's executable structure is given as Figure 4.3.

Experience gained through the development and subsequent use of the PSELECT model resulted in the decision to devote considerably greater effort to the interface between SATSUMA and the model user. The SATSUMA interface must still be described as rudimentry, but constitutes a significant increase in sophistication over PSELECT input/output mechanisms.

After the declaration of global variables and standard Pascal preparatory operations, SATSUMA proceeds via a user-interface consisting of a series of simple data input screens, prompting the user for the parameter values from which the simulated epidemic will be generated. All procedures requesting data input from the user were coded to the same general physical structure and mode of operation, and the common nomenclature, "Ask_***", was adopted to distinguish these user-interface procedures from those with a more operational function. Each Ask_*** procedure, when executed, defines a screen window displaying a message asking the user to input a variable value or series of variable values. One or a combination of custom-made functions tests the user response for conformity to the type and value of the input variable, and continues to prompt for an appropriate value until one is offered. Subsequent to acceptance by the model, this input variable is held as a global variable value for use at specified points within the program code.

As a general example, the procedure Ask_Field_Dimensions and its resultant screen is given as Figure 4.4.

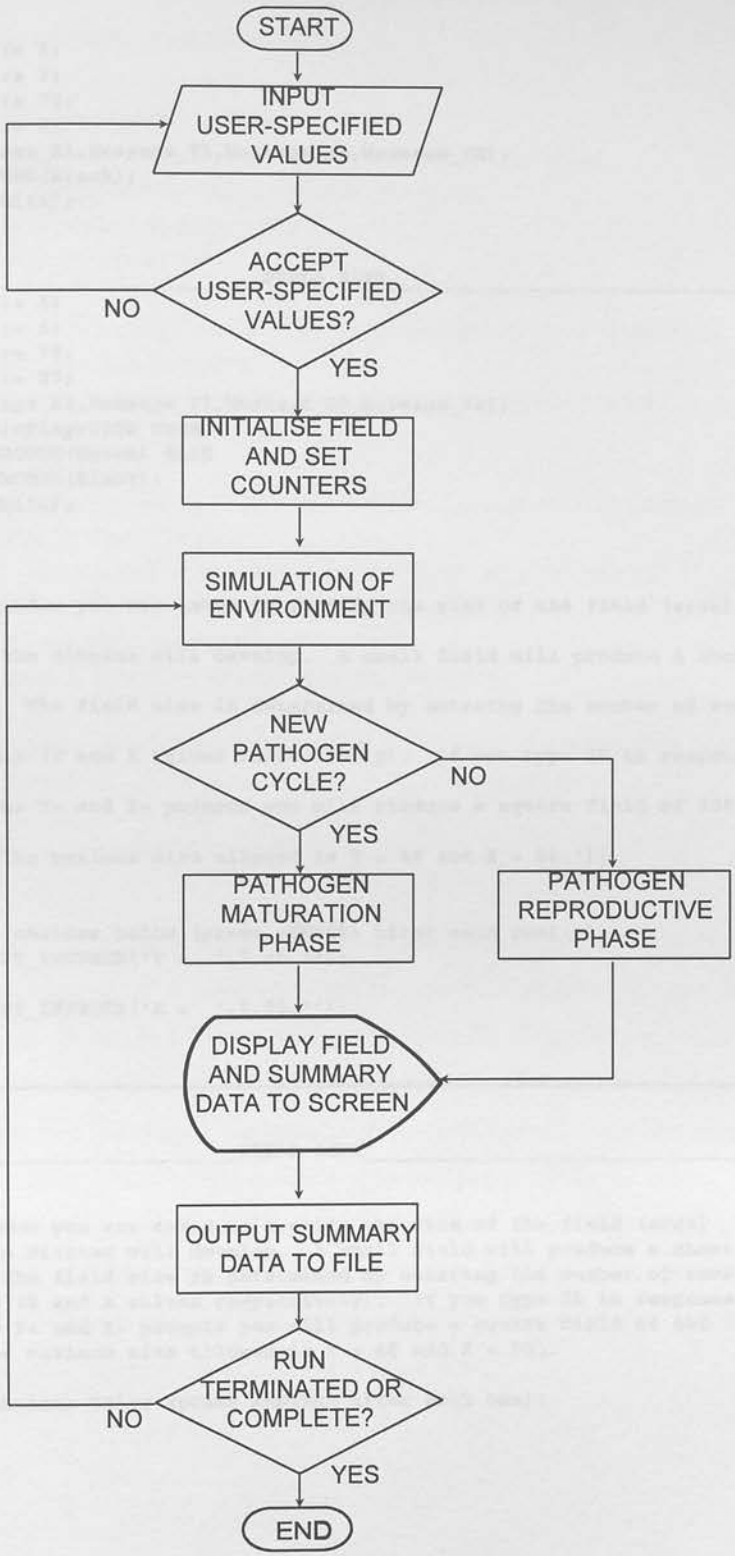


Figure 4.3. Diagrammatic representation of SATSUMA executable structure.

```
PROCEDURE Ask_Field_Dimensions;
  BEGIN
  CLRSCR;
  Message_X1 := 5;
  Message_Y1 := 3;
  Message_X2 := 75;
  Message_Y2 := 4;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITE
  ('_____FIELD SIZE_____');
  Message_X1 := 5;
  Message_Y1 := 5;
  Message_X2 := 75;
  Message_Y2 := 20;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  IF Colour_Display=TRUE THEN
    TEXTBACKGROUND(Green) ELSE
    TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITELN;
  WRITELN
  ('In this window you are asked to specify the size of the field (area)');
  WRITELN
  ('in which the disease will develop.  A small field will produce a short');
  WRITELN
  ('epidemic.  The field size is determined by entering the number of rows');
  WRITELN
  ('and columns (Y and X values respectively).  If you type 20 in response');
  WRITELN
  ('to both the Y= and X= prompts you will produce a square field of 400');
  WRITELN
  ('plants.  The maximum size allowed is Y = 40 and X = 50.');
  WRITELN;
  WRITELN
  ('Enter you choices below (press <ENTER> after each one):');
  Max_Y:=ACCEPT_INTEGER('Y =  ',1,40,'');
  WRITELN;
  Max_X:=ACCEPT_INTEGER('X =  ',1,50,'');
  END;
```

```
_____FIELD SIZE_____

In this window you are asked to specify the size of the field (area)
in which the disease will develop.  A small field will produce a short
epidemic.  The field size is determined by entering the number of rows
and columns (Y and X values respectively).  If you type 20 in response
to both the Y= and X= prompts you will produce a square field of 400
plants.  The maximum size allowed is Y = 40 and X = 50.

Enter you choices below (press <ENTER> after each one):

Y = 40

X =
```

Figure 4.4. SATSUMA procedure, Ask_Field_Dimensions, and image of the resulting user-interface screen.

User-input procedures are executed in a sequential order which depends in part upon user responses to the questions posed therein:

Screen 1: *Specification of display type*
Screen 2: *Field dimensions and position of primary infection focus*
Screen 3: *Mean Temperature and its standard deviation*
Screen 4: *Mode of pathogen dispersal*
For AIR-BORNE pathogens only:
    Screen 5: *Mean wind speed and its standard deviation*
    Screen 7: *Prevailing wind direction and whether or not it is constant*
    For variable wind direction:
        Screen 8: *The arc through which wind direction varies*
For SPLASH-DISPERSED pathogens only:
    Screen 6: *Rainfall*
    Screen 7: *Prevailing wind direction and whether or not it is constant*
    For variable wind direction:
        Screen 8: *The arc through which wind direction varies*
Screen 9: *"Noise" variable for non-specific infection failures*
Screen 10: *Plant condition character set*
Screen 11: *Displayed summary of user-specified variables*

If accepted, the user-specified variable values are input to the simulation, which then progresses to Screen 12. If rejected, Screens 1 to 11 are repeated.

Screen 12: *Filename for data output file*

All bar one Ask_*** procedures are executed in sequence within a single controlling procedure, User_Specified_Parameters. This is in turn called from within a single procedure, Accept_Specified_Parameters, responsible for controlling the input of user-specified variables into the model. This procedure holds all user-specified values in a logical "holding area", and upon acceptance of the final variable, displays a summary of the user-specified values, allowing the user either to accept the displayed values, or to repeat the input process. The procedure Ask_Output_Filename is executed separately to its related procedures in order to allow the specification of unique output file names even when the simulation is repeated using previously specified values. The operation of the user-interface routines is summarised diagrammatically as Figure 4.5.

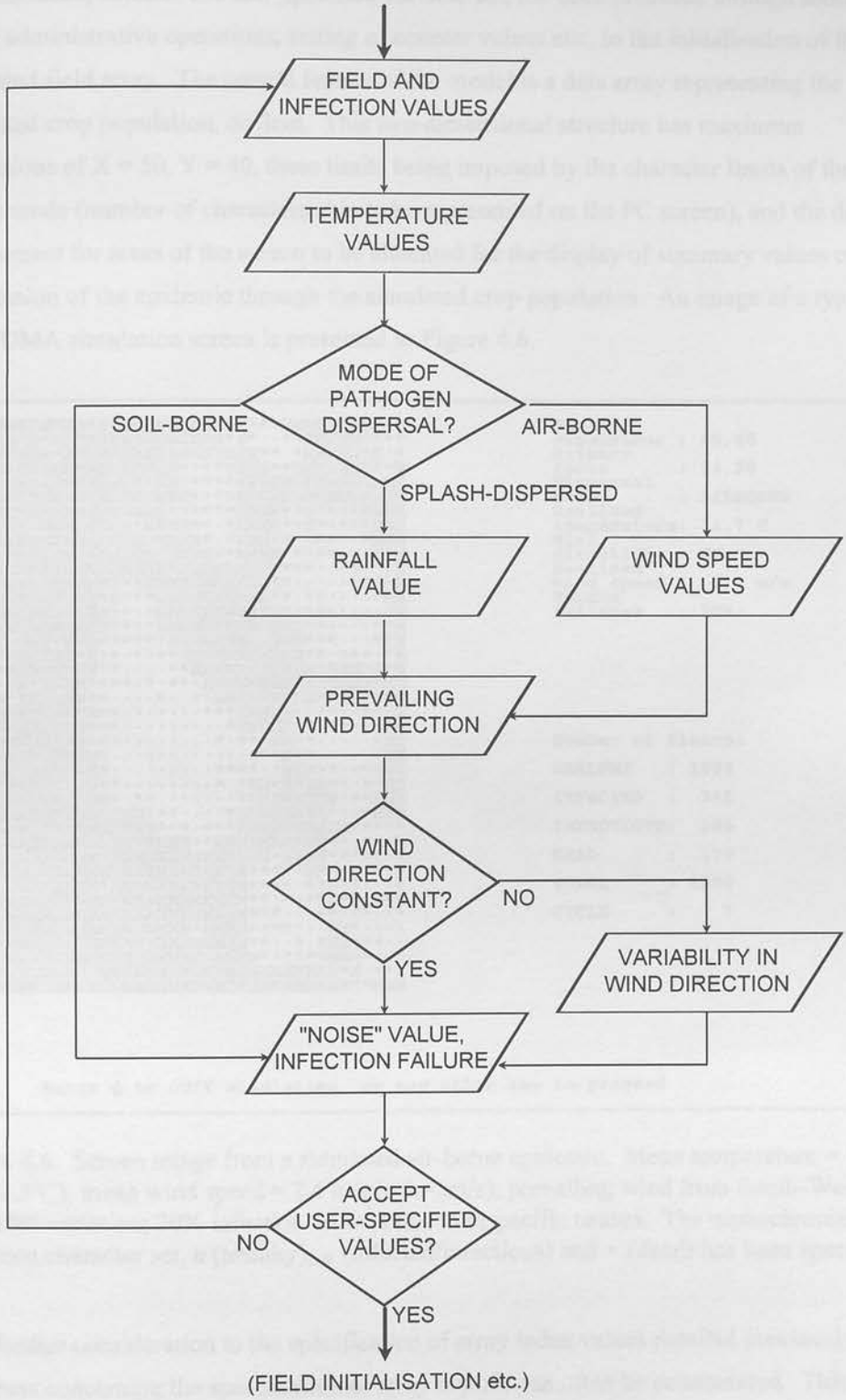


Figure 4.5. Diagrammatic representation of SATSUMA user-interface structure (equates to first two stages represented in Figure 4.3).

From the acceptance of the user-specified variable set, the code proceeds through some minor administrative operations, setting of counter values etc., to the initialisation of the simulated field array. The central feature of the model is a data array representing the simulated crop population, or field. This two-dimensional structure has maximum dimensions of X = 50, Y = 40, these limits being imposed by the character limits of the screen mode (number of characters able to be represented on the PC screen), and the design requirement for areas of the screen to be allocated for the display of summary values of the progression of the epidemic through the simulated crop population. An image of a typical SATSUMA simulation screen is presented as Figure 4.6.

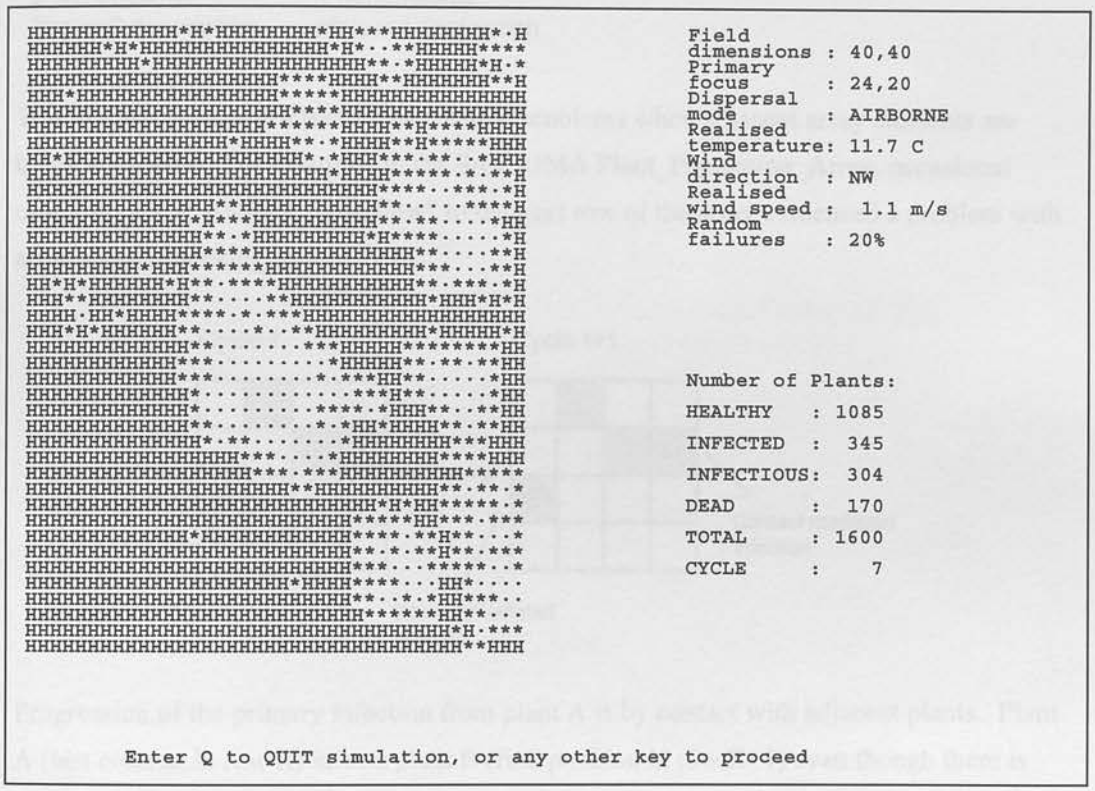
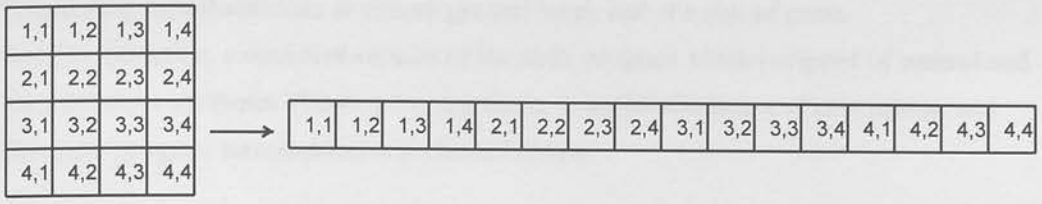


Figure 4.6. Screen image from a simulated air-borne epidemic. Mean temperature = 10°C (s.d.=2.5°C); mean wind speed = 2.4 m/s (s.d.=2m/s); prevailing wind from South-West with 180° variation; 20% infection failures due to inspecific causes. The monochrome plant condition character set, H (healthy), * (infected/infectious) and · (dead) has been specified.

As a further consideration to the specification of array index values detailed previously, problems concerning the specification of array bounds can often be encountered. This is particularly important in SATSUMA, where an array forms a central component of the model, and graphical presentation of that array is a feature of the model. If a two-dimensional (Y,X) array is specified, as with any array type, it is held within the computer memory as a single-dimension entity. For example, the 4 by 4 array below, despite being

"logically" two-dimensional, is mapped by the machine as a 16-component one-dimensional block whose two-dimensional structure is implied by the nature of its index values:

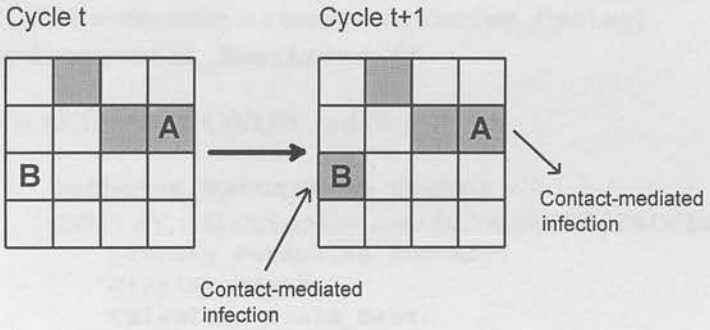
ARRAY[1..4,1..4]



2-dimensional "logical" description

1-dimensional physical description

This single-dimension structure can lead to problems when adjacent array elements are being addressed. For example, in the SATSUMA Plant_Population_Array, occasional observations of infection "overflow" to the next row of the array evidenced a problem with array bound checking.



Progression of the primary infection from plant A is by contact with adjacent plants. Plant A (last column in row R) infects plant B (first position in row R+1) even though there is apparently no contact. This can arise because the spatially remote array elements are physically adjacent in the single-dimension machine interpretation of the specified two-dimension array.

The problem was observed only at the array boundary specified by the maximum value of the X-dimension index, and was circumvented by specifying (moderately) overly large (Y+5, X+5) array indicies. Contact infections at the extreme right-hand boundary (maximum value of X) of the Plant_Population_Array are consequently trapped in the resultant "buffer" area, and lost. The problem of array overflow was not observed in alternate but equivalent situations (e.g. contact infection from plant B to plant A in the

preceding position in the array), and there was no evidence of array bound exception in the secondary (remote) infection mechanism. In the absence of evidence to the contrary, the observed array bound exceptions must be assumed to be application-specific.

Considering model structure at a more general level, and at a risk of gross oversimplification, a modified version of the main program block (stripped of control and administration attributes allowing for repetition, selective execution of procedures, and automatic program termination) is presented below.

```
[USER-INTERFACE PROCEDURES, EXECUTED AS A PRELIMINARY STAGE]
1.   Accept_Specified_Parameters; **
     Ask_Output_Filename;
     Open_Data_Output;
     Open_DBug_Output;

[INITIALISATION PROCEDURES, EXECUTED ONCE PER RUN OF THE SIMULATION:]
2.        Set_Counters;
3.        Initialise_Field;

[OPERATIONAL PROCEDURES, EXECUTED WITH EACH PASS THROUGH THE
     SIMULATION (corresponding to a complete Pathogen_Cycle):]
4.        Environmental_Routines; **

          [FOR ALTERNATE CYCLES, 5a & 5b:]

5a.            Pathogen_Maturation_Cycle;
               [DISPLAY, CALCULATION AND DATA OUTPUT PROCEDURES]
                    Display_Parameter_Summary;
                    Display_Field;
                    Calculate_Field_Data;
                    Display_Infection_Summary;
                    Output_Maturation_Summary;
               [Increment Pathogen_Cycle variable]

5b.            Fungal_Infections_Cycle; **
               [DISPLAY, CALCULATION AND DATA OUTPUT PROCEDURES]
                    Display_Parameter_Summary;
                    Display_Field;
                    Calculate_Field_Data;
                    Display_Infection_Summary;
                    Output_Infection_Summary;

     Close_Data_Output;
     Close_DBug_Output;
```

This, of course, constitutes a considerable summarisation of structural detail, the hierarchical nature of procedural programming means that procedures can be called from

within procedures ("nesting"). For example, the single reference to the Environmental_Routines procedure would be better illustrated as follows:

```
Environmental_Routines:    For all types of pathogen:
                           TEMPERATURE
                           For the case of an AIR-BORNE pathogen:
                                WIND DIRECTION
                                WIND SPEED
                           For the case of a SPLASH-DISPERSED pathogen:
                                WIND DIRECTION
                                RAINFALL
```

which in itself also constitutes an oversimplification, but in this case a more workable one. As additional illustrations of this procedural nesting, the procedure Define_Secondary_Target (which functions to limit the distribution of secondary infections) is called from within the Secondary_Infections procedure (which effects remote infections from an infectious source), which is in turn called from within the Fungal_Infections_Cycle listed in the main program block. Those procedures listed in the main program block, making internal call to other procedures, are denoted "**".

With reference to the descriptive list of operational procedures (Appendix D3), the action of key routines can be described. Following from the acceptance of user-specified variable values into the model via the Ask_*** procedures, all counter values are set to 1 or 0 as appropriate. A healthy population array of dimensions Y_Max by X_Max is established, and an infected plant established at position Y0, X0.

For each complete cycle through the simulation, corresponding to a single increment of the Pathogen_Cycle variable, the underlying activity (potential to cause infection, or degenerate to successive plant conditions) P_g, is determined by the Temperature_Effect procedure. The value of mode of pathogen dispersal is then assessed to determine the executable flow through subsequent procedures. For air-borne pathogens, Realised_WindDirection (obtained from the specified prevailing wind direction and the potential variation about that direction) is used to determine the pattern of primary (contact) infection and the potential area of secondary (remote) infections that can be initiated by an assessed infectious plant. Realised_WindSpeed (from specified Mean_WindSpeed and its standard deviation) determines the potential infective range for primary infection. For splash-dispersed pathogens, wind direction procedures are executed as described for air-borne pathogens, but wind speed is not taken into account. Instead, the specified level of rainfall directly determines the potential ranges for both primary and secondary infection.

For alternate passes through the program code, corresponding to the assessed value of the Cycle_Number variable (cf. Pathogen_Cycle, each increment of which corresponds to two increments of Cycle_Number), pathogen infectious and maturation phases are simulated. A phase-specific series of screen-control procedures - displaying a summary of key environmental parameters, running totals for the plant states within the population, and the central plant population array itself - are executed. The infectious status of plants within the simulated population is assessed, and crop population data output to file.

Alternation between infection and maturation phases is effected through assessment of the Cycle_Number variable. The coding structure,

```
IF ((Cycle_Number MOD 2) = 1) THEN
```

determines whether the current value of Cycle_Number is wholly divisible by 2. If so, SATSUMA pathogen maturation phase procedures are executed, and if not, infection routines are run. An interesting feature of this arrangement is that, although the pathogen maturation procedures physically preceded those for infection, the executable order is infection followed by maturation (1 is an odd number, and hence fails the "*MOD 2*" test).

All non-administrative procedures are executed in a sequential order which depends on the user-specified values entered into the model via the user-interface procedures. For the sake of brevity, the full listing of SATSUMA program code is listed as Appendix D2. A technical discussion of individual procedures is presented as Appendix D3.

VALIDATION & VERIFICATION

Analysis of simulated epidemics

Principal requirements from the model output are twofold. It is important to establish that: (1) the model output is suitable for interpretation by users of a range of abilities, from a very qualitative treatment ("what does the map look like"), to a more advanced quantitative treatment; (2) the model can be subjected to sensitivity analysis to illustrate the effects of the key environmental parameters influencing a crop disease epidemic.

In order to illustrate the robustness of the model and its utility as an introduction to methods of temporal and spatial analysis, a fixed set of user-specified variables were used to generate replicated disease progress curves. The following user-specified variables were input to the model:

1. Field dimensions = 40 by 40 plants
2. Primary infection focus specified non-randomly (plant at position Y=20, X=20)
3. Mean temperature = 19°C, with standard deviation = 5.0
4. For air-borne/splash-dispersal simulation, prevailing wind from the South-West with no variation
5. For air-borne simulation, mean wind speed = 5.0 ms^{-1}, with standard deviation = 5.0
   For splash-dispersal simulation, rainfall is typical.
6. Threshold velocity for spore release was discounted
7. Probability of infection failure due to non-specified causes = 5%

Temporal analysis

It was essential to demonstrate that SATSUMA output produces accepted/generalised disease progress curves, which are commonly taken as the starting point for the introductory description of plant disease epidemics. Because of the stochastic nature of SATSUMA, it was necessary to demonstrate that the range of disease progress curves generated by a given set of input values would be sufficiently homogeneous to avoid the occurrence of misleading disease progress curves which reduce the value of the model as a teaching tool. Furthermore, for more advanced courses of study, if projects are to be developed in which course participants fit a mathematical model to the output from the simulation, it is necessary to be confident that the output will be accurately summarised by one of the commonly used functions from the plant pathology literature. Temporal development of the

epidemic was therefore analysed by non-linear regression of the number of dead plants against time to verify that this was the case.

In order to examine the variation in the parameters of the curves non-linear regression analysis was conducted using the Genstat V (VMS version, release 2.1) statistical package (NAG Ltd., 1987). Genstat analysis was conducted by Neil McRoberts, SAC Auchincruive. The logistic and Gompertz functions are among the most commonly used in plant disease epidemiology to describe sigmoid curves of disease development (Berger, 1981; Campbell & Madden, 1990). A number of variations of the logistic and Gompertz functions can be found in the literature. In Genstat the functions are given in the following formats:

logistic; $y = A+(C/(1+\exp(-B(t-M))))+e$

Gompertz; $y = A+C(\exp(-\exp(-B(t-M))))+e$

In both functions t is time (number of complete pathogen cycles), A is the lower asymptote (basal), A+C is the upper asymptote (maximal), M is the value of t at the point of inflection (defined as being where the value of the second differential, d^2y/dt^2, equals 0), B is the slope parameter at the point of inflection (Figure 4.7), and e is the residual (error) term. In the current model y, the level of disease, is given by the number of dead plants.

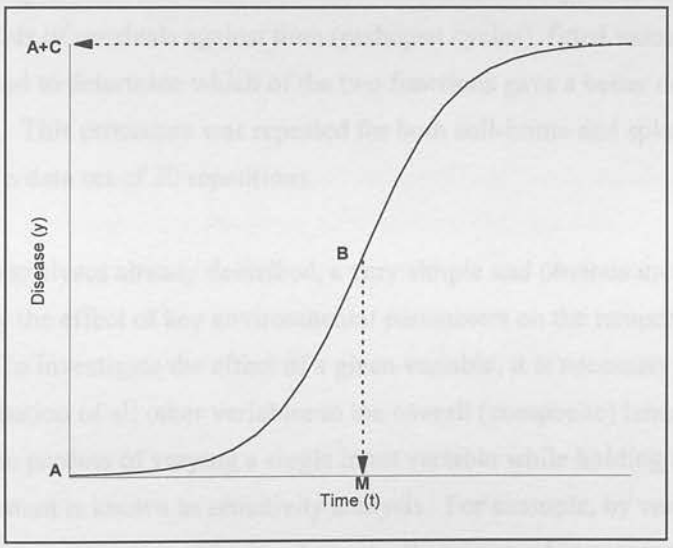


Figure 4.7. Graphical description of parameters estimated by non-linear regression analysis of simulated epidemics.

The decision as to which function best fits the data cannot be made on the basis of the coefficient of determination, R^2, (sometimes referred to as "the percentage variance accounted for - %VAF"), alone, because the cumulative nature of a disease progress curve leads to high R^2 values even if an inappropriate model is used (Campbell & Madden, 1990). In order to establish which model best estimates the simulated disease progress curves it is therefore necessary to consider not only the R^2 value. Owing to this fact, a number of standard methods which examine the residuals from the estimated curve can be used: (1) plotting of the residuals against time in order to establish an absence of systematic deviation of observed values against the fitted curve; (2) plotting of the residuals against the fitted values of the disease progress curve in order to detect a polarity of deviation from the fitted curve; (3) calculation of time-lagged residuals (residual at t against residual at t-1) in order to establish an absence of correlation between successive residual values, as a further check that there is no systematic deviation of the estimated curve from the observed data.

The SATSUMA air-borne simulation was repeated 100 times using the variable values previously listed. Results of number of dead plants against time (cycles) were fed into an Excel spreadsheet, number of cycles to epidemic completion was calculated for 100 repetitions and was then used to sort the results of the repetitions into ascending order. In order to make a preliminary decision about which function should be used to examine the complete data set, the logistic and Gompertz functions were fitted to a sample of 10 curves comprising the longest, the shortest, and 8 others chosen arbitrarily. The coefficients of determination, plots of residuals against time (pathogen cycles), fitted values and lagged residuals were used to determine which of the two functions gave a better description of the generated curves. This procedure was repeated for both soil-borne and splash-dispersed epidemics, using a data set of 20 repetitions.

In addition to the analyses already described, a very simple and obvious use of the model would be to study the effect of key environmental parameters on the temporal progression of an epidemic. To investigate the effect of a given variable, it is necessary to discount or negate the contribution of all other variables to the overall (composite) temporal profile of the epidemic. The process of varying a single input variable while holding all other sources of variability constant is known as sensitivity analysis. For example, by varying the temperature regime, epidemic profiles can be mutually compared to assess the effect of different temperatures. It should be noted that the results obtained in this way are valid only for that particular set of environmental constants, and that if a different set of constant values were defined, the effect of the variable under consideration might change.

For a true sensitivity analysis to be conducted all sources of variation, other than that under consideration, must be eliminated. In the case of a stochastic model, internal sources of variation (i.e. the generation and use of random deviates) must therefore be removed or circumvented. The simplest way to do this is to specify a random seed applicable to each random procedure within the model. The string of pseudo-random deviates so generated will still appear random within itself, but will be entirely constant among repetitions. This action does, however, render the model strictly deterministic, and care must be taken in drawing inferences about the function of the stochastic model from the results of sensitivity analysis conducted on a deterministic version of that model.

Spatial pattern analysis

As discussed previously, a number of methods of spatial pattern analysis are available for examining the distribution of diseased plants in a crop. The main concern to date has been to ensure that the model produces data which can be analysed by the commonly used methods. A printed map of the simulated field can be produced at each cycle of the epidemic, on which spatial pattern analyses can be conducted. Current methodology falls into three categories: (1) Frequency distribution; (2) Distance class analysis; (3) Geostatistical methods. Each methodology presents specific advantages and disadvantages, and all methods can be conducted on the screen output from the model. The first two of these methods were applied to SATSUMA output.

(1) Frequency distribution analysis (FDA)

Frequency distribution analysis was conducted using the BBD program (Madden & Hughes, 1994). In order to obtain the data for fitting of frequency distributions, the outer two rows of the map were ignored in order to eliminate edge-effects, and the remainder of the map was divided into a grid of contiguous square quadrats, each containing 9 plants (3 x 3). In each quadrat the number of diseased (= infected, infectious, or dead) plants was recorded. BBD output provides the user with an estimate of the goodness of fit of the observed number of dead plants per quadrat to both the binomial and beta-binomial distributions. In addition, for the estimated beta-binomial distribution the parameters, θ, the degree of aggregation, and p, the mean probability of a given plant being infected, are also generated. Spatial information about the field under analysis is not preserved in this method of analysis.

Frequency distribution analysis is a numerical method, and its advantages and disadvantages largely reflect this. In terms of its advantages, output from FDA is easily interpreted with respect to the output from its rival methods; aggregation information is captured in a small number of parameters (sometimes a single parameter) which is useful for further applications (e.g. sampling); output from the analysis is numerical and hence compatible with temporal analysis results, and therefore does not preclude the development of combined spatio-temporal analysis models. In terms of disadvantage, fitting a frequency distribution to incidence data (e.g. incidence of diseased plants per quadrat) will determine between a random or aggregated distribution, the data themselves do not yield specific location data to describe the distribution because the spatial data is lost in the translation of graphical data to a series of numerical values. Methods of spatial analysis are, in general, computationally demanding, although the recent availability of computer software to conduct methods has to an extent negated this consideration.

(2) Distance class analysis (DCA)

Two-dimensional distance class analysis was performed using the program 2DCLASS (Nelson *et al.*, 1992). If the field is considered as a two-dimensional lattice of plants, plant (i,j) is the ith plant in the jth row, then two-dimensional distance-class analysis is summarised by reference to i and j. The distance between infected plants is defined by both horizontal (i) and vertical (j) units. Pairs of infected plants are thus grouped into two-dimensional (i,j) distance classes that refer to the absolute distance between plants in a pair. Because the total number of possible pairs varies between (i,j) distance classes, the number of pairs of infected plants in each (i,j) distance class is standardized by the total number of pairs of living plants occurring within the same (i,j) distance class. This allows for direct comparison of standardized count frequency (SCF) values in any (i,j) distance classes. The standardised number of pairs of infected plants in each (i,j) distance class is compared directly to expected values obtained from up to 400 computer-generated simulations of a lattice containing the same number of infected plants arranged under the assumption of a random pattern. A significance level on the observed SCF for each (i,j) distance class is computed directly by counting the number of times the simulated SCF exceeds the observed SCF during the 400 simulations. This procedure allows the detection of a departure from randomness, and preserves the two-dimensional spatial information needed to discern cluster size and arrangement (Campbell & Madden, 1990).

With specific reference to the 2DCLASS method, DCA provides very accurate information about parameters describing the level of aggregation of disease within the population (e.g.

the number and size of clusters). In addition to this accuracy, DCA is a highly visual method, with all the inherent advantages thereof. Disadvantages arise through the nature of the output. While the visual output is very descriptive, it is not intuitively interpreted. The output requires considerable experience to avoid the drawing of erroneous conclusions. As an additional consideration, the output is not in a format readily compatible with empirical methods used in temporal analysis.

By way of a comparative summary, the advantages and disadvantages of the respective methods depend in large part upon the intended application (Hughes & Nelson, 1994). Intensive mapping methods (e.g. 2DCLASS) are more suited to studies of pathosystem ecology because they take explicit account of the location of diseased plants. Sparse sampling methods (e.g. BBD) are, however, better for summarising spatial patterns for subsequent application in disease management.

Model output

As a qualitative illustration of the progression of a simulated epidemic in SATSUMA, Figures 4.8a to 4.8d show four consecutive population arrays generated by the model. The primary infection focus was randomly positioned (Y = 24, X = 20) within the field array. Prevailing wind was from the South-West, but variation through 180° was permitted. Figure 4.8a shows the epidemic after the maturation phase of pathogen cycle 4. Infection is clustered primarily around the primary infection focus, but two significant secondary infection foci have been established downwind from the primary infection focus, along with some minor infection foci towards the South-Eastern corner of the field. The simulated epidemic developed to the point that at the end of the pathogen cycle 7 maturation phase infection was widespread, but still predominantly situated within the Eastern sector of the simulated field.

Figures 4.8a-4.8d were obtained from the model running in monochrome display mode, with indistinguishability between infected and infectious plant units specified. Plant conditions are represented by the characters `H` (healthy), `*` (infected/infectious) and `·` (dead). A different character set is employed if colour display mode is specified by the user.

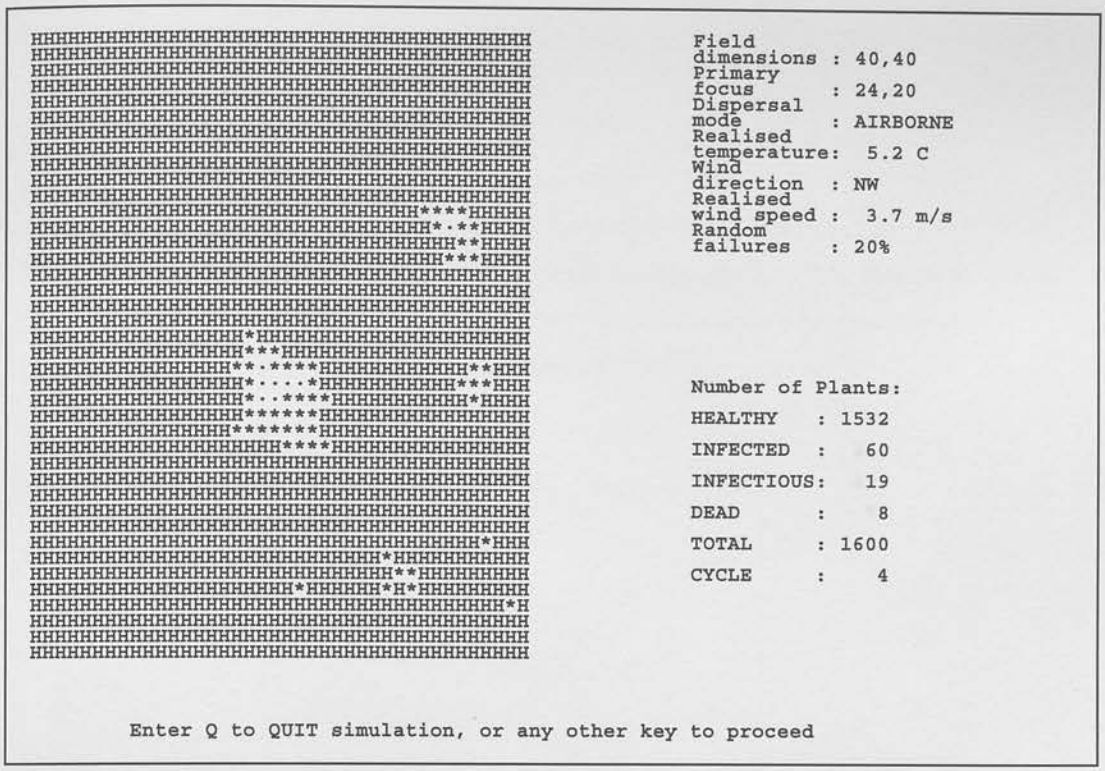


Figure 4.8a. Cycle 4 (maturation phase) of a SATSUMA air-borne epidemic.

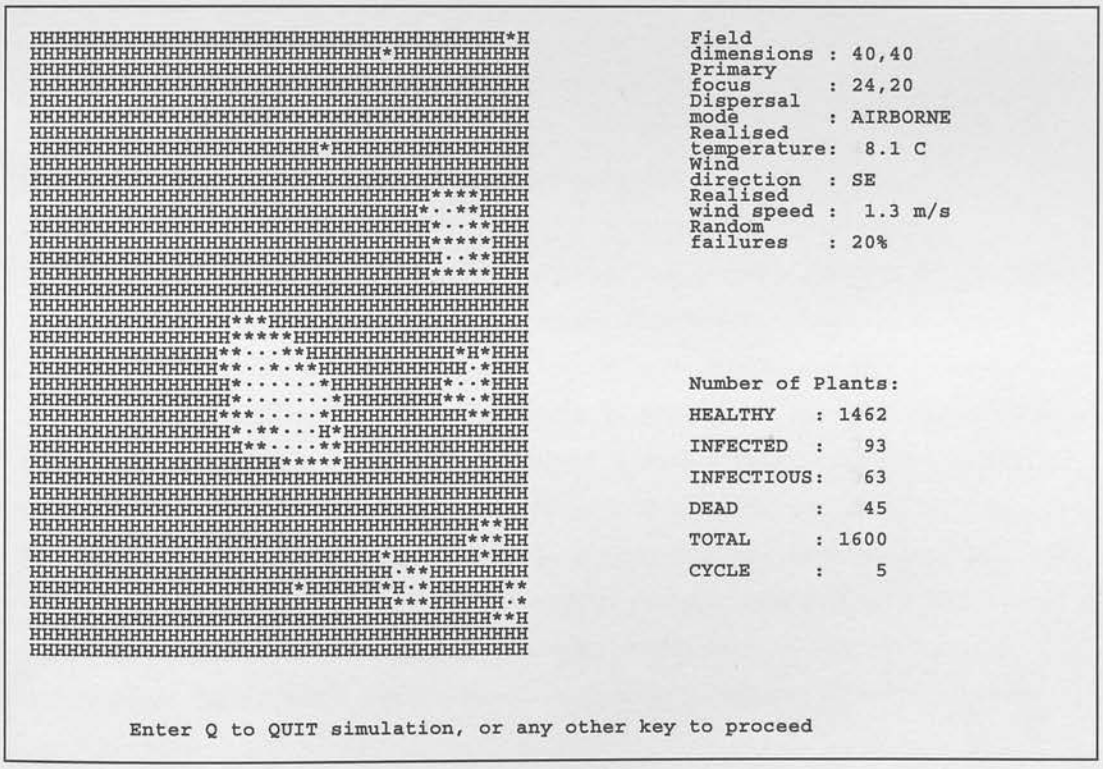


Figure 4.8b. Cycle 5 (maturation phase) of a SATSUMA air-borne epidemic.

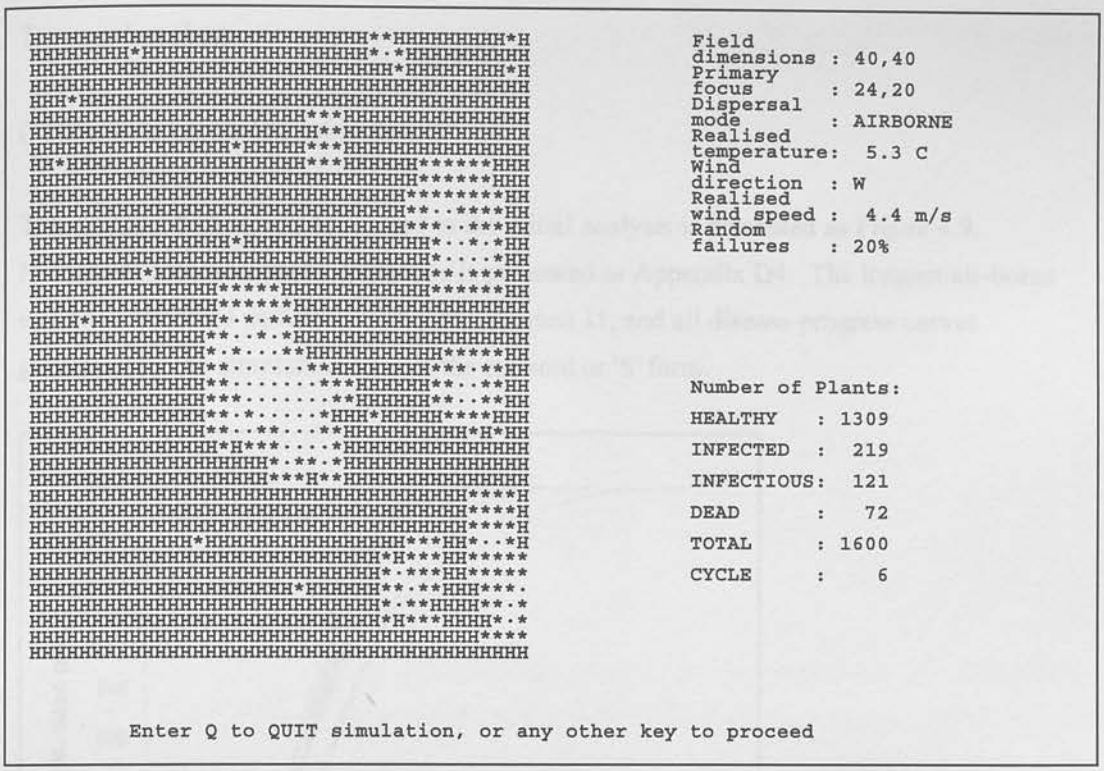


Figure 4.8c. Cycle 6 (maturation phase) of a SATSUMA air-borne epidemic.

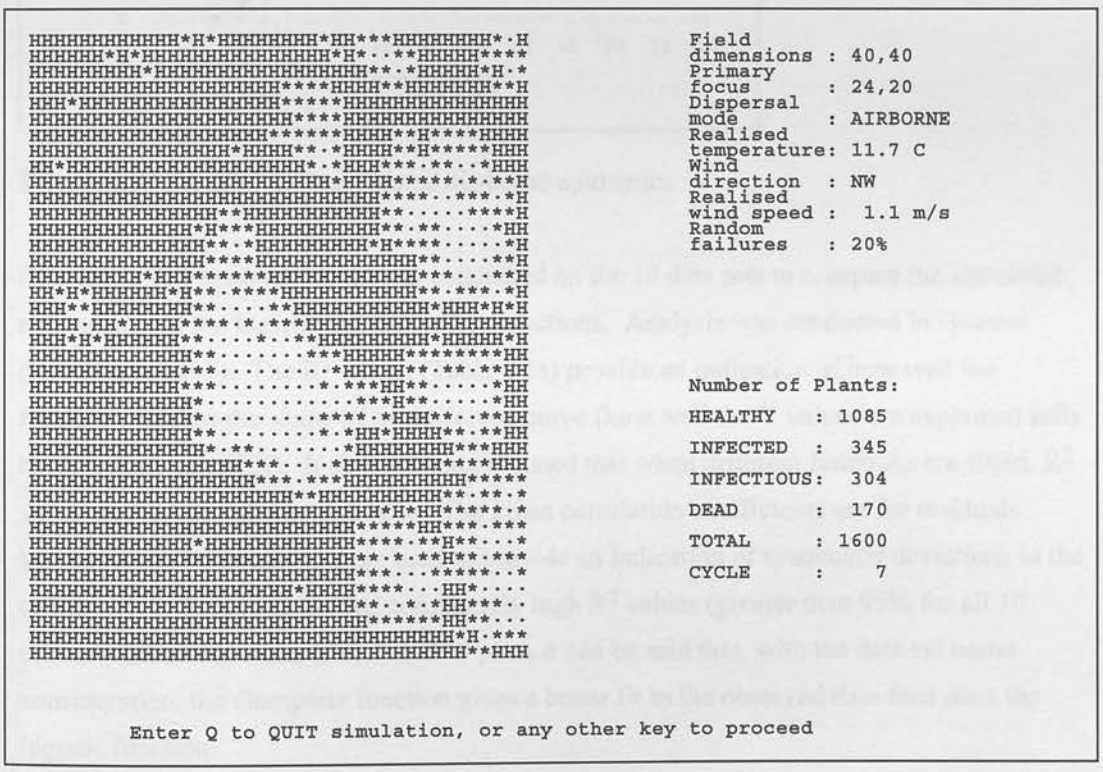


Figure 4.8d. Cycle 7 (maturation phase) of a SATSUMA air-borne epidemic.

(1) Simulated air-borne epidemic

The sample of 10 curves to be used in the initial analysis is presented as Figure 4.9. Numerical data for sample epidemics is presented as Appendix D4. The longest air-borne epidemic lasted 23 pathogen cycles, the shortest 11, and all disease progress curves generated by the simulation were of the sigmoid or 'S' form.

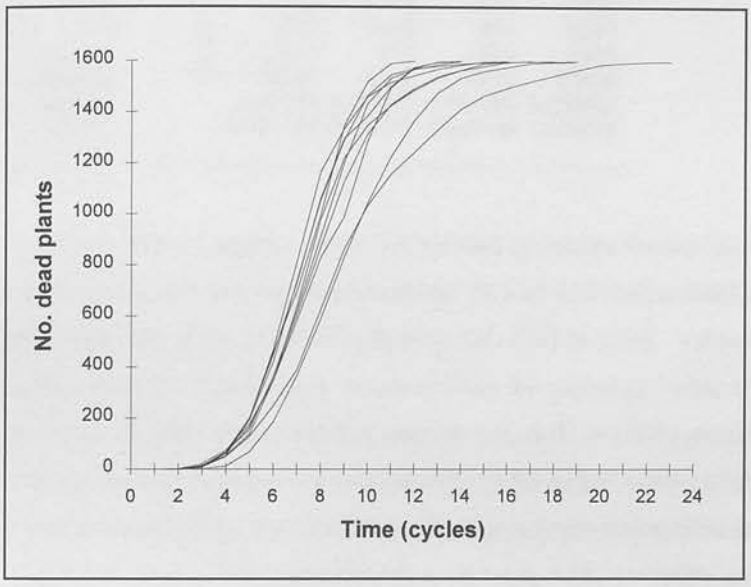


Figure 4.9. Sample of 10 simulated air-borne epidemics

Non-linear regression analysis was conducted on the 10 data sets to compare the simulated results against the logistic and Gompertz functions. Analysis was conducted in Genstat (NAG Ltd., 1987). The R^2 values (Table 4.1a) provide an indication of how well the function matches the shape of the observed curve (how well the Y values are explained fully by their function of X). It should be appreciated that when different functions are fitted, R^2 values are not directly comparable. The given correlation coefficients are for residuals versus lagged residuals, and are used to provide an indication of systematic deviations in the curves. Both functions yielded consistently high R^2 values (greater than 95% for all 10 curves), but on the basis of the residual plots it can be said that, with the data set under consideration, the Gompertz function gives a better fit to the observed data than does the logistic function.

Table 4.1a. R^2 values from the regression and correlation coefficients for the residuals and lagged residuals.

Epidemic no.	no. of cycles	logistic function		Gompertz function	
		R**2	cor. coeff.	R**2	cor. coeff.
1	11	99.9	0.467	99.9	0.032
7	12	99.8	0.114	99.5	0.163
2	14	99.9	0.057	99.7	0.509
3	14	99.6	0.6	99.6	0.214
8	14	99.9	0.275	99.9	0.06
4	17	100	0.651	99.9	0.542
9	17	99.8	0.606	99.9	0.225
5	19	100	0.234	99.9	0.537
10	19	99.9	0.576	99.9	0.695
6	23	99.7	0.71	99.9	0.441
Means	**16**	**99.85**	**0.429**	**99.81**	**0.3418**
St dev.		**0.12693**	**0.238332**	**0.152388**	**0.230473**
s.e.m.		**0.040139**	**0.075367**	**0.048189**	**0.072882**

Analysis of the Genstat output file yielded estimates for the key parameters describing the epidemic: lower and upper asymptotes (A and A+C respectively), mean rate of epidemic (B), and time at the point of inflection (M) (Table 4.1b). Values for epidemic start and end points from the fitted curve obtained from the values in Table 4.1b, equivalent to the first and last real data points (in this case, A = 0, A+C = 1600) provide an indication of how closely the estimated curve corresponds to the actual values (Table 4.1c). In fitting a function to the data, the aim is to minimise any deviation between the estimated and observed values. No constraints (e.g. A >= 0, A+C <= 1600) were applied in the parameter estimation.

Table 4.1b. Parameter estimates from the regression analysis.

Epidemic no.	no. of cycles	logistic function				Gompertz function			
		A	B	C	M	A	B	C	M
1	11	-30.6	0.9015	1606.2	6.9096	11.18	0.6274	1589.1	6.3124
7	12	-23.9	0.8045	1691.5	7.3305	12.4	0.4961	1769.5	6.74
2	14	-18.9	0.7951	1636.3	7.5868	20.7	0.5258	1654.3	6.9334
3	14	-42.8	0.6403	1711.4	8.026	7.8	0.4123	1757	7.307
8	14	-38.7	0.7865	1632.8	7.1418	8.7	0.5302	1637.1	6.5072
4	17	-38.18	0.6936	1625	7.672	15.1	0.4821	1606.3	6.9451
9	17	-43.2	0.7433	1620.9	6.4207	28.2	0.537	1572.3	5.7952
5	19	-26.28	0.8421	1624.2	7.1763	18.7	0.5988	1595	6.5278
10	19	-44.8	0.6036	1644.4	8.8686	6.36	0.41738	1631.4	8.0247
6	23	-91.4	0.4667	1664.2	8.699	1.1	0.34431	1598	7.7985
Mean	**16**	**-39.876**	**0.72772**	**1645.69**	**7.58313**	**13.024**	**0.497139**	**1641**	**6.88913**
s.d.	**3.681787**	**20.19061**	**0.128996**	**33.46452**	**0.769606**	**7.903067**	**0.08689**	**68.98865**	**0.676934**
s.e.m.	**1.164283**	**6.384832**	**0.040792**	**10.58241**	**0.243371**	**2.499169**	**0.027477**	**21.81613**	**0.214065**

Table 4.1c. Fitted values of the start and end-points of the epidemics, obtained from the logistic and Gompertz functions.

Epidemic no.	no. of cycles	logistic function start	logistic function end	Gompertz function start	Gompertz function end
1	11	-22.8	1575.2	11.2	1596.6
7	12	-13.6	1629	12.4	1656.3
2	14	-10.2	1607.5	20.7	1635.2
3	14	-23.9	1632.1	7.8	1657
8	14	-25.7	1586.7	8.7	1615.2
4	17	-22.4	1584.3	15.1	1608.8
9	17	-15	1577	28	1597
5	19	-17	1598	19	1613
10	19	-30.7	1596	6.4	1621.2
6	23	-46.8	1570.7	1.1	1590.5
Mean		**-22.81**	**1595.65**	**13.04**	**1619.08**
St dev.		**10.46725**	**21.56217**	**7.880947**	**23.6995**
s.e.m.		**3.310034**	**6.818557**	**2.492174**	**7.494439**

Gompertz curves have been fitted to a wide range of observed epidemics, in crops as diverse as celery and maize (Waggoner, 1986). Having determined that the Gompertz function gave the best fit to the data, that function was then fitted to the complete data set to give 100 estimates of A, A+C, B and M. From these estimates the mean, sample standard deviation and standard error of the mean values were derived (Table 4.2), and these values used to calculate 95% confidence intervals for that mean to give higher and lower predicted values for the parameters. MathCad software (MathSoft Inc.) was then used to generate the curves defined by these parameter values, which were then qualitatively compared with disease profiles generated by SATSUMA (Figure 4.10).

Table 4.2. Population parameter estimates, derived from fit of Gompertz function to complete data set.

		B	M	C	A
Air-borne epidemic, Gompertz model (n=100)	**Mean**	0.50	5.91	1627.97	16.30
	s.d.	0.07	0.48	42.40	12.11
	s.e.m.	0.01	0.05	4.24	1.21
	Lower 95% c.i.	0.36	4.95	1543.17	-7.91
	Upper 95% c.i.	0.64	6.87	1712.77	40.52

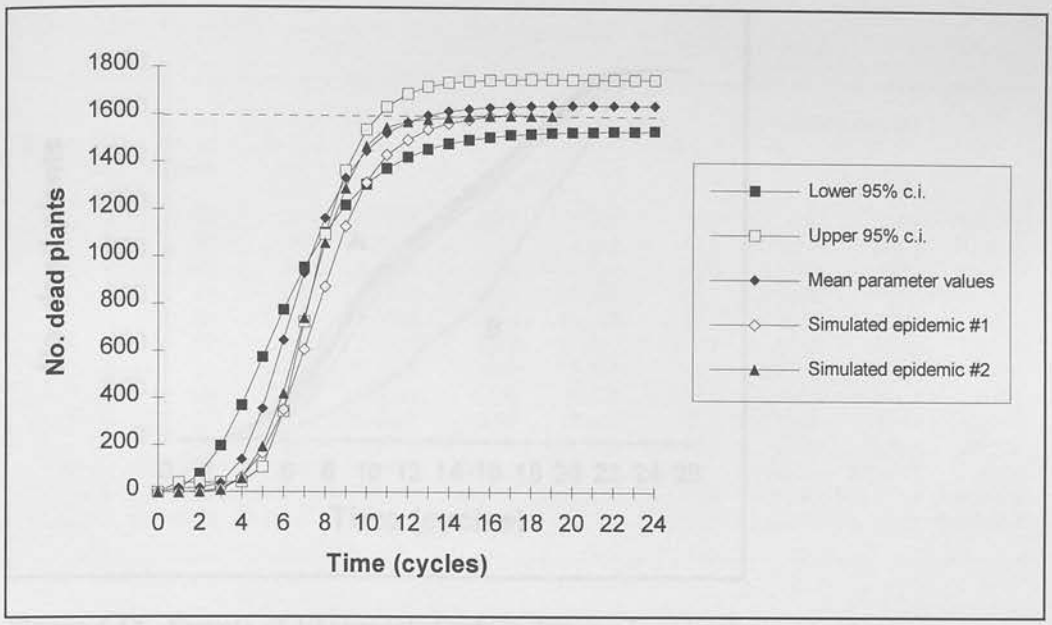


Figure 4.10. Plot of parameter estimates (mean, upper and lower 95% c.i., Gompertz function), for comparison with simulated air-borne epidemics.

(2) Simulated splash-dispersed epidemic

The splash-dispersed epidemics (Figure 4.11) were similar in length to the soil-borne ones. The shortest was 23 cycles, the longest 26. The difference between splash and air modules of the program seem quite clear (the shortest air-borne epidemic was only 11 cycles). R^2 values, correlation coefficients, parameter estimates and fitted start/end-point values (Tables 4.3a-4.3c) were calculated as described for the simulated air-borne epidemics. Neither the logistic nor the Gompertz model was entirely satisfactory for splash-dispersed epidemics due to the very high correlation between lagged residuals. It's a curiosity that the splash-dispersal model should fit so poorly relative to either of the other models. Both the logistic and Gompertz models produced fairly close approximations to the actual data and on the basis that it had slightly less bad correlation between residuals, the logistic model was fitted (Table 4.4). Curves defined by these parameter values were qualitatively compared with disease profiles generated by SATSUMA (Figure 4.12).

The response curves for splash-dispersed epidemics (Figure 4.11) show two intersting features. A noticeable edge-of-field effect is seen at approximately cycle 10 (A), where the pattern of infection changes from contact and spatially remote (but proximal) infections to predominantly contact infection. One epidemic (B) also presents a nice illustration of extreme-case consequences of the stochastic nature of the SATSUMA model.

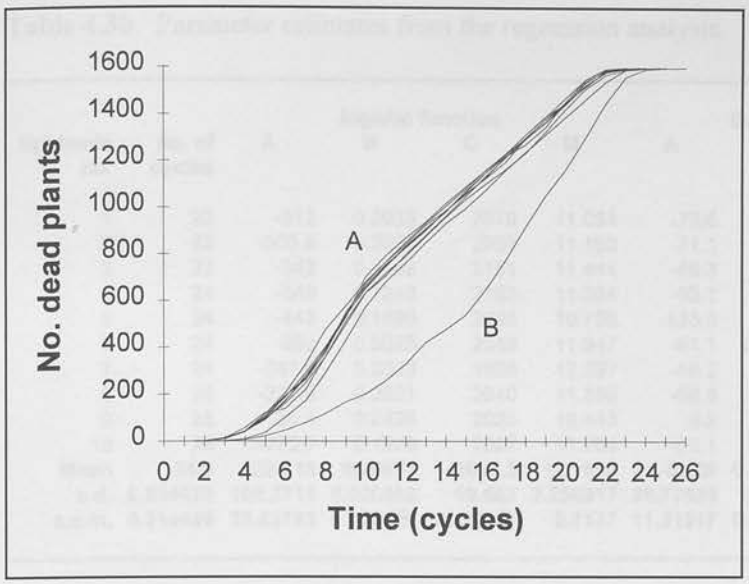


Figure 4.11. Sample of 10 simulated splash-dispersed epidemics.

Table 4.3a. R^2 values from the regression and correlation coefficients for the residuals and lagged residuals.

Epidemic no.	no. of cycles	logistic function		Gompertz function	
		R**2	cor. coeff.	R**2	cor. coeff.
1	23	99.3	0.719	99.5	0.747
2	23	99.5	0.718	99.7	0.736
3	23	99.4	0.69	99.5	0.718
4	24	99.5	0.595	99.6	0.62
5	24	99.4	0.578	99.5	0.617
6	24	99.6	0.581	99.7	0.57
7	24	99.3	0.762	99.6	0.783
8	25	99.6	0.563	99.7	0.569
9	25	99.8	0.353	99.6	0.456
10	26	99.4	0.624	99.5	0.643
Mean		**99.48**	**0.6183**	**99.59**	**0.6459**
s.d.		**0.154919**	**0.116859**	**0.08756**	**0.100877**
s.e.m.		**0.04899**	**0.036954**	**0.027689**	**0.0319**

Table 4.3b. Parameter estimates from the regression analysis.

Epidemic no.	no. of cycles	logistic function A	B	C	M	Gompertz function A	B	C	M
1	23	-312	0.2033	2070	11.054	-72.6	0.1566	1898	10.074
2	23	-300.6	0.2033	2052	11.162	-71.1	0.1538	1902.2	10.169
3	23	-343	0.1869	2161	11.444	-88.3	0.1423	1989	10.453
4	24	-349	0.1843	2163	11.384	-93.1	0.1399	1933	10.396
5	24	-448	0.1689	2288	10.736	-133.5	0.1327	2045	9.906
6	24	-263	0.2025	2049	11.947	-61.1	0.14719	1951.8	10.845
7	24	-247.6	0.2086	1998	12.397	-58.2	0.1533	1898.8	11.292
8	25	-292.5	0.2021	2040	11.286	-68.8	0.1527	1891.7	10.232
9	25	-30.4	0.2496	2025	18.443	9.2	0.1149	2755	18.836
10	26	-275.7	0.1978	2027	11.786	-59.1	0.1489	1885.4	10.666
Mean	**24.1**	**-286.18**	**0.20073**	**2087.3**	**12.1639**	**-69.66**	**0.144229**	**2014.99**	**11.2869**
s.d.	**0.994429**	**106.3715**	**0.020953**	**89.557**	**2.256917**	**35.77539**	**0.01267**	**264.9546**	**2.683268**
s.e.m.	**0.314466**	**33.63763**	**0.006626**	**28.32041**	**0.7137**	**11.31317**	**0.004007**	**83.78601**	**0.848524**

Table 4.3c. Fitted values of the start and end-points of the epidemics, obtained from the logistic and Gompertz functions.

Epidemic no.	no. of cycles	logistic function start	end	Gompertz function start	end
1	23	-75.1	1590	-42.3	1590.9
2	23	-69.8	1581.5	-39.5	1584.3
3	23	-74.4	1594.3	-45.6	1594
4	24	-71.3	1621	-45	1623.6
5	24	-78	1619.5	-55	1619.7
6	24	-61.7	1621.6	-33.6	1628.4
7	24	-77.9	1587.3	-42.5	1590.6
8	25	-65.8	1627.4	-37.2	1634.5
9	25	-4.7	1664.6	10.4	1693.2
10	26	-61.2	1636.1	-31.4	1643.5
Mean		**-63.99**	**1614.33**	**-36.17**	**1620.27**
s.d.		**21.70604**	**26.02879**	**17.66786**	**33.10804**
s.e.m.		**6.864052**	**8.231026**	**5.587069**	**10.46968**

Table 4.4. Population parameter estimates, derived from fit of logistic function to complete data set.

		B	M	C	A
Splash-dispersed epidemic, logistic model (n=20)	**Mean**	0.19	11.67	2131.55	-326.22
	s.d.	0.02	2.10	94.63	107.09
	s.e.m.	0.00	0.47	21.16	23.95
	Lower 95% c.i.	0.18	10.69	2087.22	-376.39
	Upper 95% c.i.	0.20	12.66	2175.88	-276.05

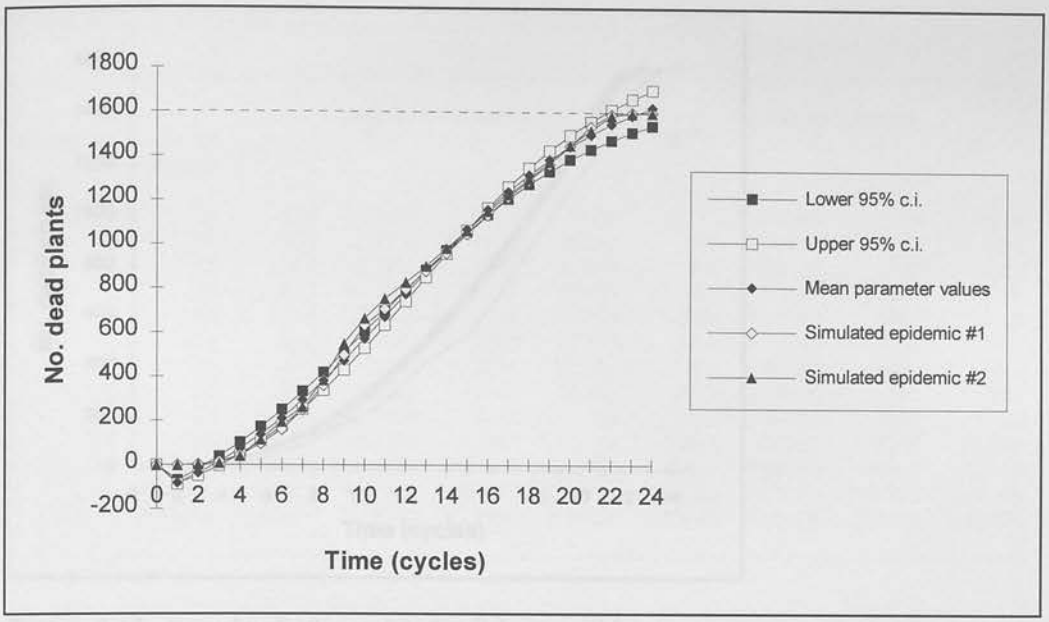


Figure 4.12. Plot of parameter estimates (mean, upper and lower 95% c.i., logistic function), for comparison with simulated splash-dispersed epidemics.

(3) Simulated soil-borne epidemic

The epidemics were far more uniform than were the air-borne simulations (Figure 4.13). This is probably related to the fact that the RANDOM function is called only once each cycle to determine the temperature-dependent growth rate, whereas in the air-borne model the function is also called for determining the secondary infections etc. The shortest epidemic lasted 22 cycles, the longest 25. Analysis of 10 of the epidemics (longest, shortest + 8 others) with logistic and Gompertz models (Tables 4.5a-4.5c) indicated that the logistic curve gave a more satisfactory fit, and this function was fitted to the full data set (Table 4.6). The decision was much clearer in this case than for the air-borne type, mainly due to the lower correlation between residuals and lagged residuals for the logistic model, than for the Gompertz model. Curves defined by the parameter values were then qualitatively compared with disease profiles generated by SATSUMA (Figure 4.14).

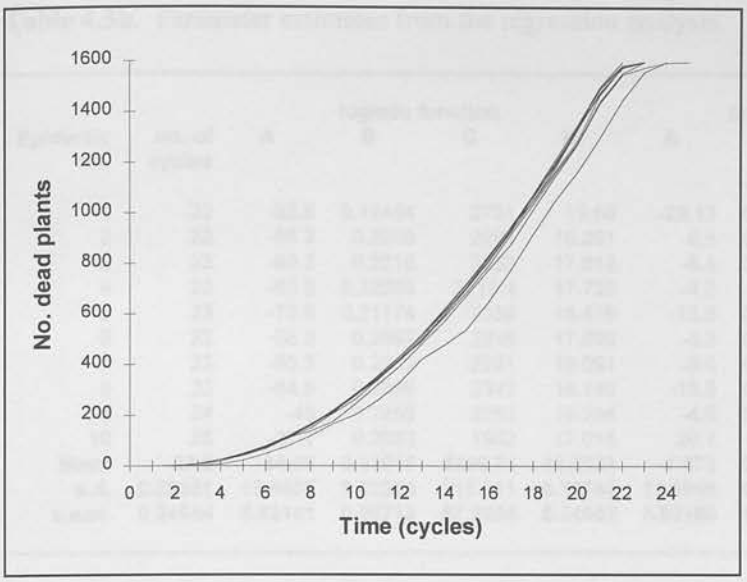


Figure 4.13. Sample of 10 simulated soil-borne epidemics.

Table 4.5a. R^2 values from the regression and correlation coefficients for the residuals and lagged residuals.

Epidemic no.	no. of cycles	logistic function		Gompertz function	
		R**2	cor. coeff.	R**2	cor. coeff.
1	22	99.9	0.172	100	0.033
2	23	99.9	0.025	99.8	0.126
3	23	99.9	0.007	99.8	0.063
4	23	99.9	0.063	99.9	0.126
5	23	99.9	0.077	99.9	0.078
6	23	99.9	0.058	99.8	0.185
7	23	99.9	0.025	99.8	0.126
8	23	99.9	0.049	99.9	0.13
9	24	99.9	0.111	99.8	0.231
10	25	99.7	0.519	99.5	0.6
Mean		**99.88**	**0.1106**	**99.82**	**0.1698**
s.d.		**0.06325**	**0.15126**	**0.13166**	**0.16157**
s.e.m.		**0.02**	**0.04783**	**0.04163**	**0.05109**

Table 4.5b. Parameter estimates from the regression analysis.

Epidemic no.	no. of cycles	logistic function A	B	C	M	Gompertz function A	B	C	M
1	22	-92.6	0.19454	2791	19.68	-29.13	0.07568	5174	23.76
2	23	-65.3	0.2209	2291	18.091	-8.5	0.09762	3403	19.5
3	23	-69.3	0.2216	2263	17.812	-8.5	0.10138	3222	18.821
4	23	-60.9	0.22656	2219.4	17.725	-4.2	0.1027	3181	18.76
5	23	-75.6	0.21174	2359	18.476	-13.6	0.09361	3541	20.06
6	23	-58.2	0.2667	2246	17.893	-4.3	0.10042	3307	19.17
7	23	-65.3	0.2209	2291	18.091	-8.5	0.09762	3403	19.5
8	23	-84.5	0.2096	2343	18.145	-15.5	0.09675	3340	19.252
9	24	-48	0.2258	2282	19.394	-4.6	0.09302	3685	21.54
10	25	-27.1	0.2633	1902	17.016	20.1	0.1365	2287	16.432
Mean	**23.2**	**-64.68**	**0.22616**	**2298.74**	**18.2323**	**-7.673**	**0.09953**	**3454.3**	**19.6795**
s.d.	**0.78881**	**18.4405**	**0.02256**	**215.011**	**0.78746**	**12.3066**	**0.01506**	**710.976**	**1.91173**
s.e.m.	**0.24944**	**5.83141**	**0.00713**	**67.9924**	**0.24902**	**3.89168**	**0.00476**	**224.83**	**0.60454**

Table 4.5c. Fitted values of the start and end-points of the epidemics, obtained from the logistic and Gompertz functions.

Epidemic no.	no. of cycles	logistic function start	end	Gompertz function start	end
1	22	-20.8	1612.8	-10	1621.1
2	23	-13.9	1647	-0.7	1664.3
3	23	-16	1649.2	-1.2	1665.9
4	23	-11.8	1642.9	2.3	1661.5
5	23	-18.7	1629.4	-4.4	1643.4
6	23	-10.4	1650.8	2.4	1669.9
7	23	-13.9	1647	-0.7	1664.3
8	23	-21.8	1636.8	-5.8	1649.8
9	24	-12.8	1638.4	-0.3	1658.4
10	25	0.5	1667.7	20.7	1697
Mean		**-13.96**	**1642.2**	**0.23**	**1659.56**
s.d.		**6.34791**	**14.4974**	**8.1276**	**19.5348**
s.e.m.		**2.00739**	**4.58449**	**2.57017**	**6.17745**

Table 4.6. Population parameter estimates, derived from fit of logistic function to complete data set.

		B	M	C	A
Soil-borne epidemic, logistic model (n=20)	**Mean**	0.23	17.88	2206.54	-59.22
	s.d.	0.02	0.85	234.10	18.36
	s.e.m.	0.00	0.19	52.35	4.11
	Lower 95% c.i.	0.22	17.48	2096.88	-67.82
	Upper 95% c.i.	0.24	18.28	2316.20	-50.62

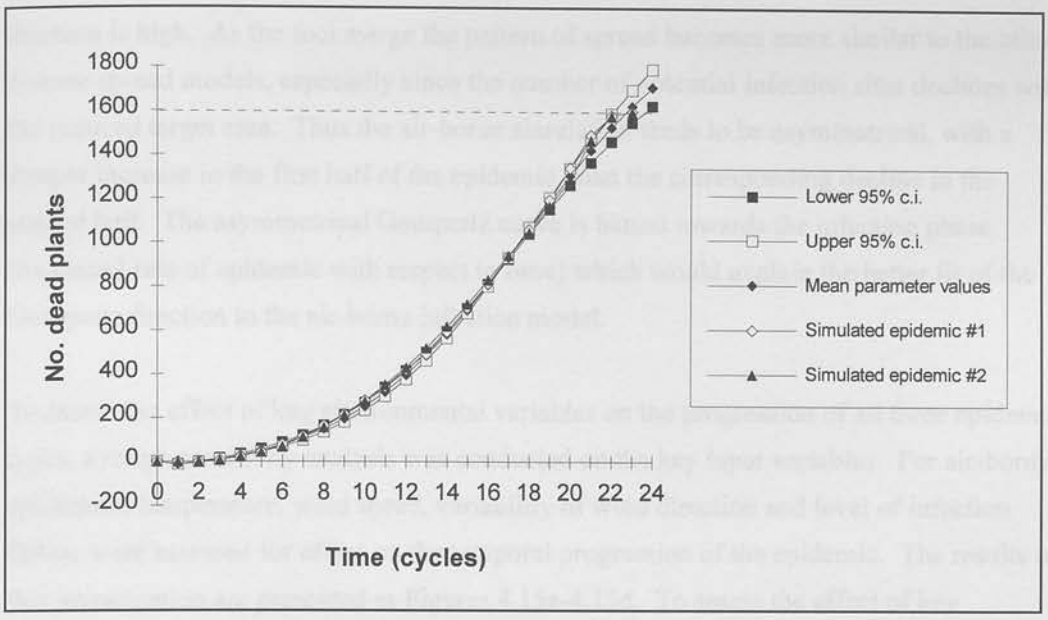


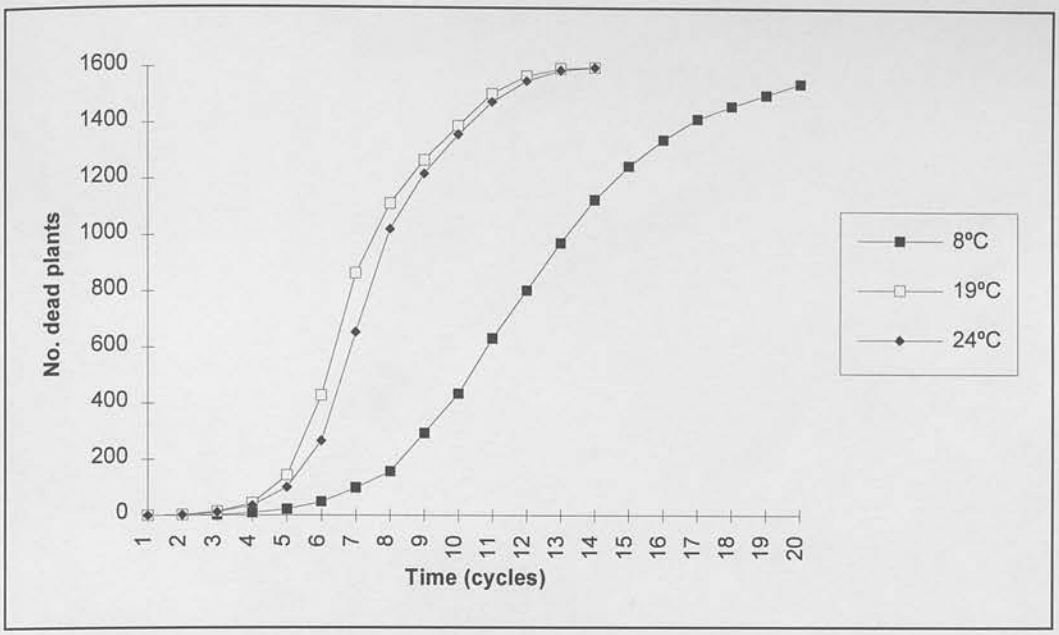
Figure 4.14. Plot of parameter estimates (mean, upper and lower 95% c.i., logistic function), for comparison with simulated soil-borne epidemics.

It was observed that the Gompertz function gave a better fit to the air-borne model, while a better fit for the soil-borne and splash-dispersed models was obtained through the logistic function. There is no hard and fast rule about which mathematical function (model) will fit a particular set of data. Campbell & Madden (1990) give a good worked example of four potato varieties in the same potato blight trial. For three varieties the Gompertz function fitted better, while for the other the logistic model was superior. In addition to this, neither the logistic model nor the Gompertz model were originally formulated for use as descriptive functions for plant disease epidemics, and there is no *a priori* reason why either should give an adequate description of observational data. The fact that SATSUMA seems to violate some of the underlying assumptions of both functions does not mean that SATSUMA is invalid as a simulator of plant disease epidemics. Other methods of analysis are available for situations where consecutive disease values are correlated in space or time (e.g. Repeated Measures methodologies). Going back to the original aims of the simulation as a teaching tool, it is sufficient to demonstrate that SATSUMA gives sigmoid curves. The complexities of how these should be analysed are beyond the scope of this study.

The logistic curve is symmetrical. In the soil and splash models the spread of the pathogen is either exclusively (soil) or predominantly (splash) from a single infection focus. Thus the increase in the number of dead plants follows a geometrically determined path which will be approximately symmetrical. The air-borne simulator quickly initiates a number of infection foci, which are independent of the primary infection focus. Hence the initial rate of disease

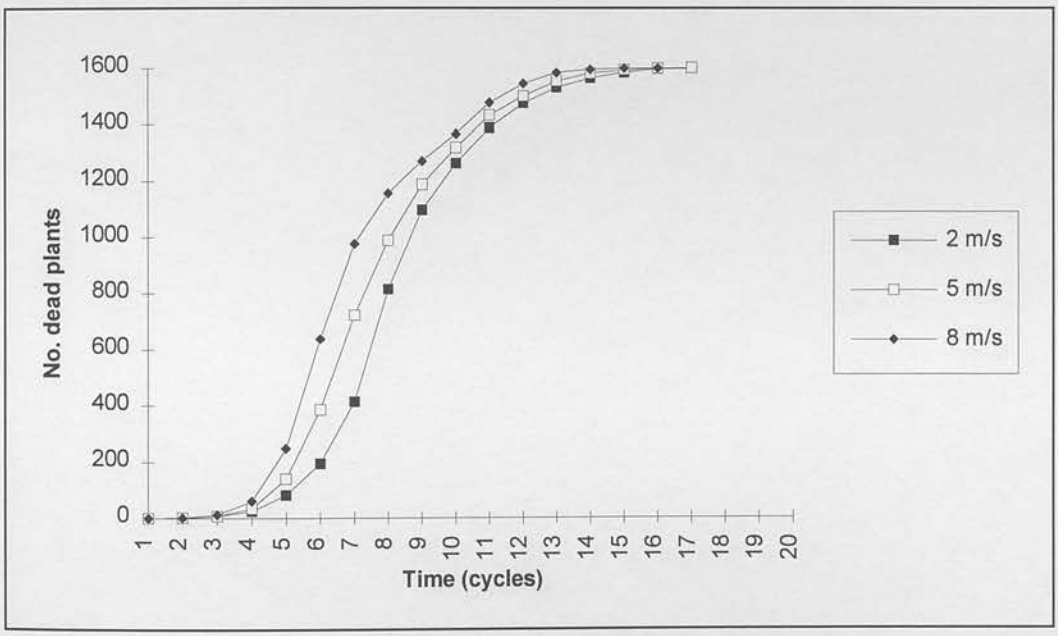
increase is high. As the foci merge the pattern of spread becomes more similar to the other disease spread models, especially since the number of potential infection sites declines with the reduced target area. Thus the air-borne simulation tends to be asymmetrical, with a steeper increase in the first half of the epidemic, than the corresponding decline in the second half. The asymmetrical Gompertz curve is biased towards the infection phase (increased rate of epidemic with respect to time) which would explain the better fit of the Gompertz function to the air-borne infection model.

To assess the effect of key environmental variables on the progression of all three epidemic types, a rough sensitivity analysis was conducted on the key input variables. For air-borne epidemics, temperature, wind speed, variability of wind direction and level of infection failure were assessed for effect on the temporal progression of the epidemic. The results of this investigation are presented as Figures 4.15a-4.15d. To assess the effect of key environmental variables on the progression of a splash-dispersed epidemic, sensitivity analysis was conducted on the input variables: temperature, rainfall, variability of wind direction and level of infection failure. The results of the investigation are presented as Figures 4.16a-4.16d. To assess the effect of key environmental variables on the progression of a soil-borne epidemic, sensitivity analysis was conducted on temperature and level of infection failure. The results from this investigation are presented as Figures 4.17a & 4.17b. Sensitivity analysis of the key environmental variables, for all three modes of pathogen dispersal, yielded results consistent with the design of the model.



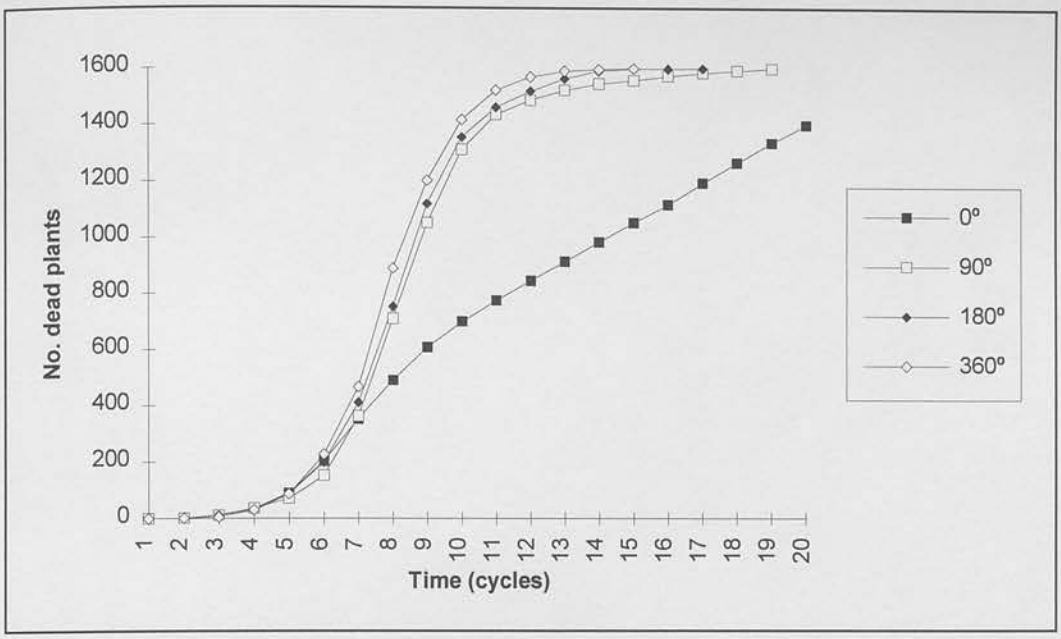
Field = 40x40 = 1600 plants; primary infection focus at Y=10, X=10; prevailing wind from the North-West, with no variation in direction; wind speed = 3.0 m/s, σ = 0; 100% infection success.

Figure 4.15a. The effect of temperature on a simulated air-borne epidemic.



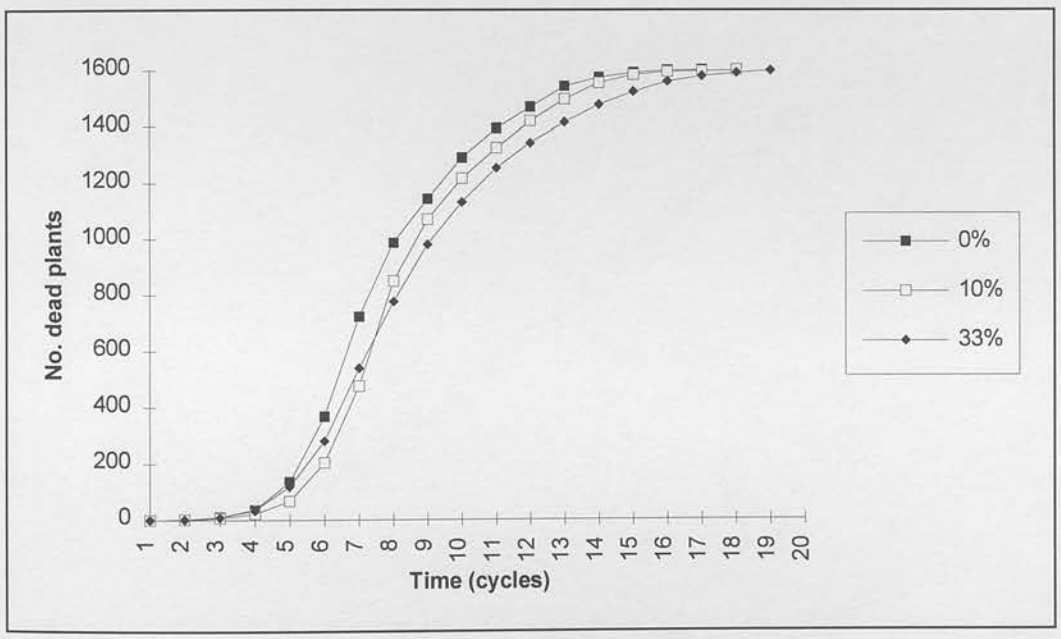
Field = 40x40 = 1600 plants; primary infection focus at Y=10, X=10; temperature = 12.0°C, σ = 0; prevailing wind from the North-West, with no variation in direction; 100% infection success.

Figure 4.15b. The effect of wind speed on a simulated air-borne epidemic.



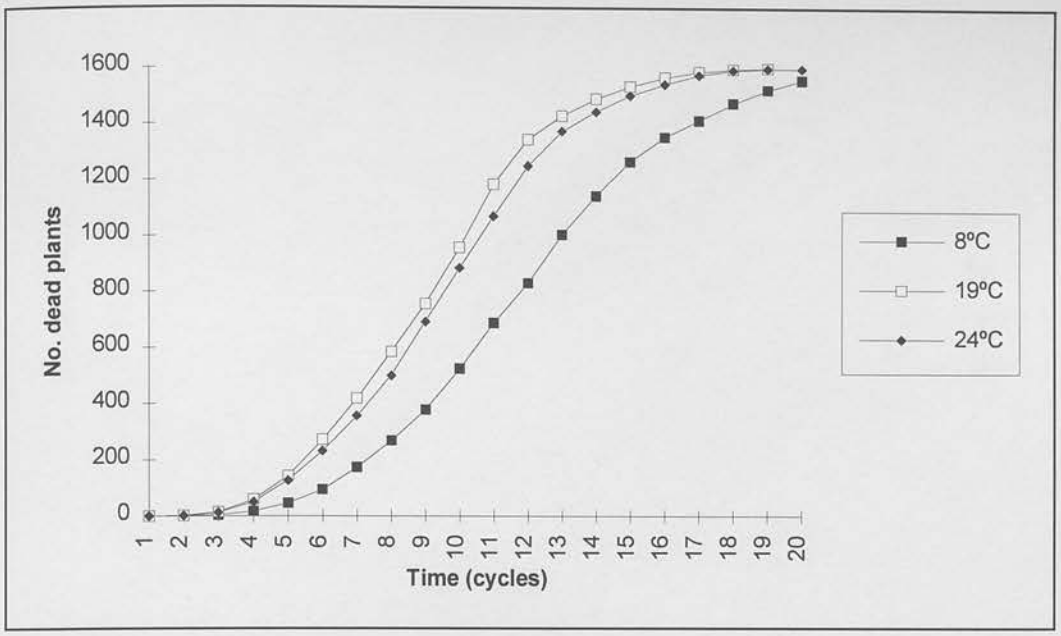
Field = 40x40 = 1600 plants; primary infection focus at Y=20, X=20; temperature = 12.0°C, σ = 0; prevailing wind from the North-West, with wind speed = 5.0 m/s, σ = 0; 100% infection success.

Figure 4.15c. The effect of wind direction variability on a simulated air-borne epidemic.



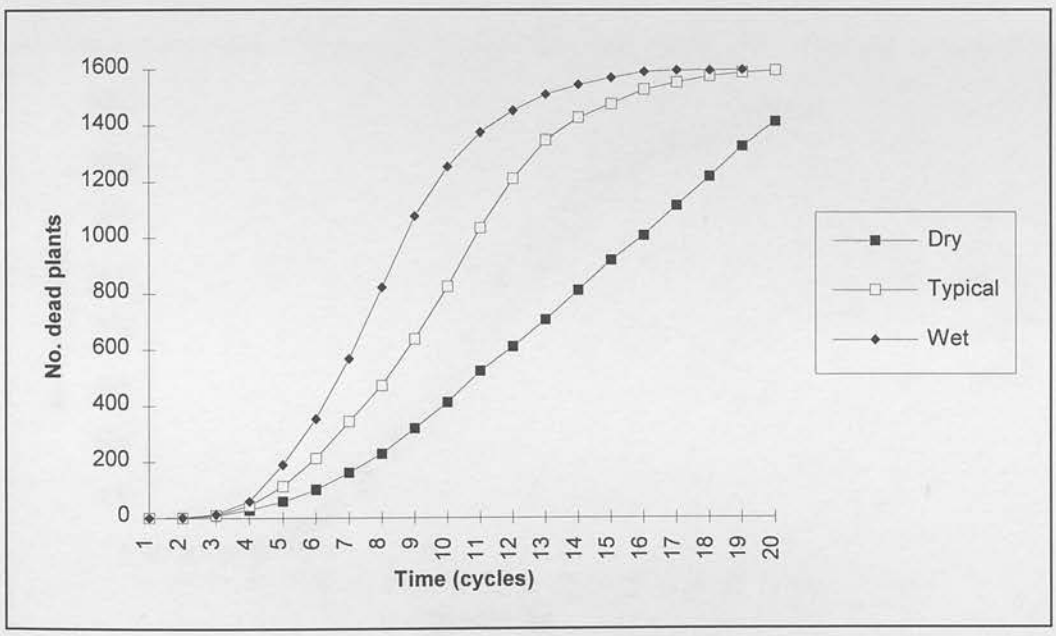
Field = 40x40 = 1600 plants; primary infection focus at Y=10, X=10; temperature = 12.0°C, σ = 0; wind speed = 5.0 m/s, σ =0; prevailing wind from the North-West, with no variation in direction; 100% infection success.

Figure 4.15d. The effect of inspecific infection failures on a simulated air-borne epidemic.



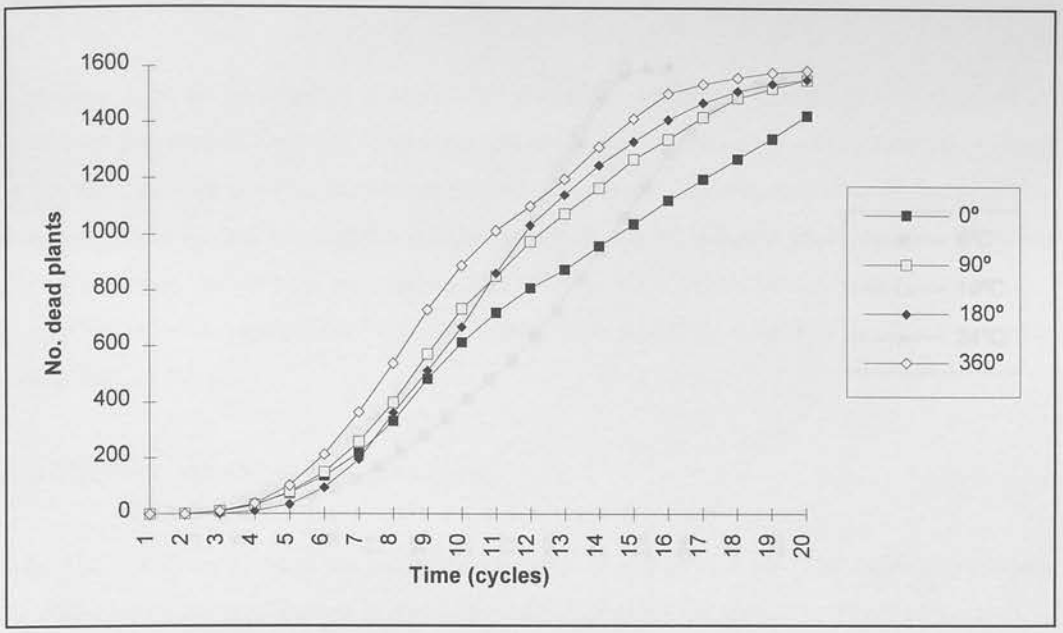
Field = 40 X 40 = 1600 plants; primary infection focus at Y=10, X=10; rainfall is typical; prevailing wind from the North-West, with no variation in direction; 100% infection success.

Figure 4.16a. The effect of temperature on a simulated splash-dispersed epidemic.



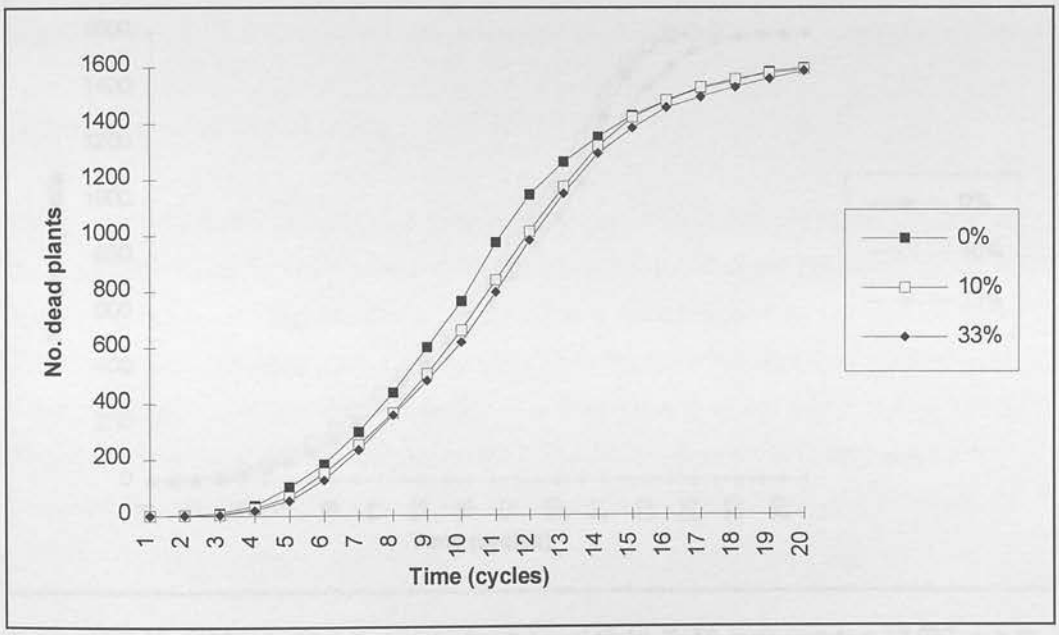
Field = 40 X 40 = 1600 plants; primary infection focus at Y=10, X=10; temperature = 12.0ºC, σ = 0; prevailing wind from the North-West, with no variation in direction; 100% infection success.

Figure 4.16b. The effect of rainfall on a simulated splash-dispersed epidemic.



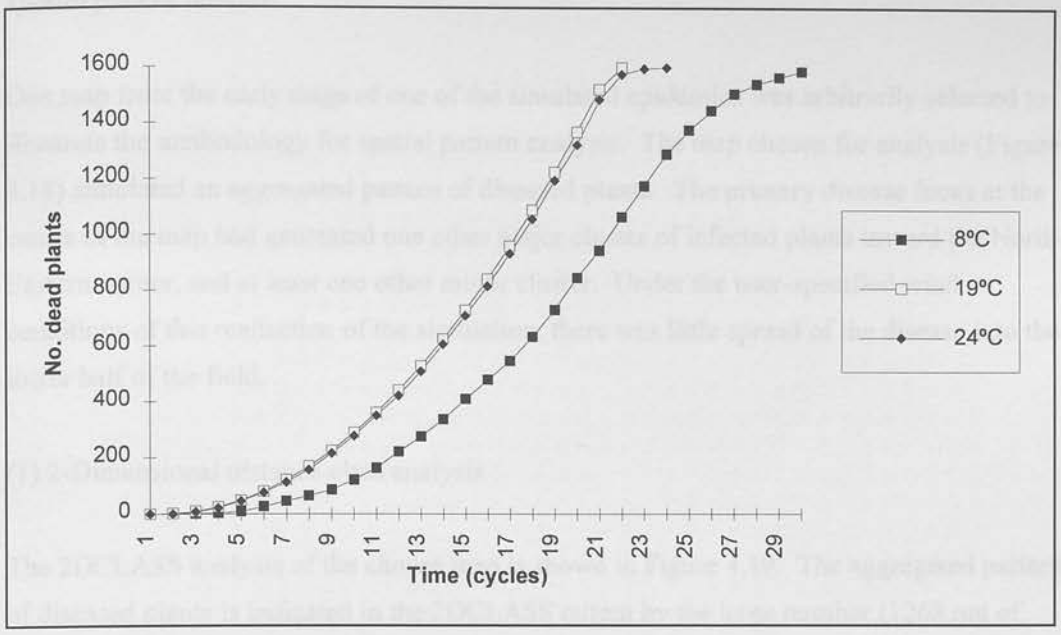
Field = 40 X 40 = 1600 plants; primary infection focus at Y=20, X=20; temperature = 12.0°C, σ = 0; rainfall is typical; prevailing wind from the North-West; 100% infection success.

Figure 4.16c. The effect of wind direction variability on a simulated splash-dispersed epidemic.



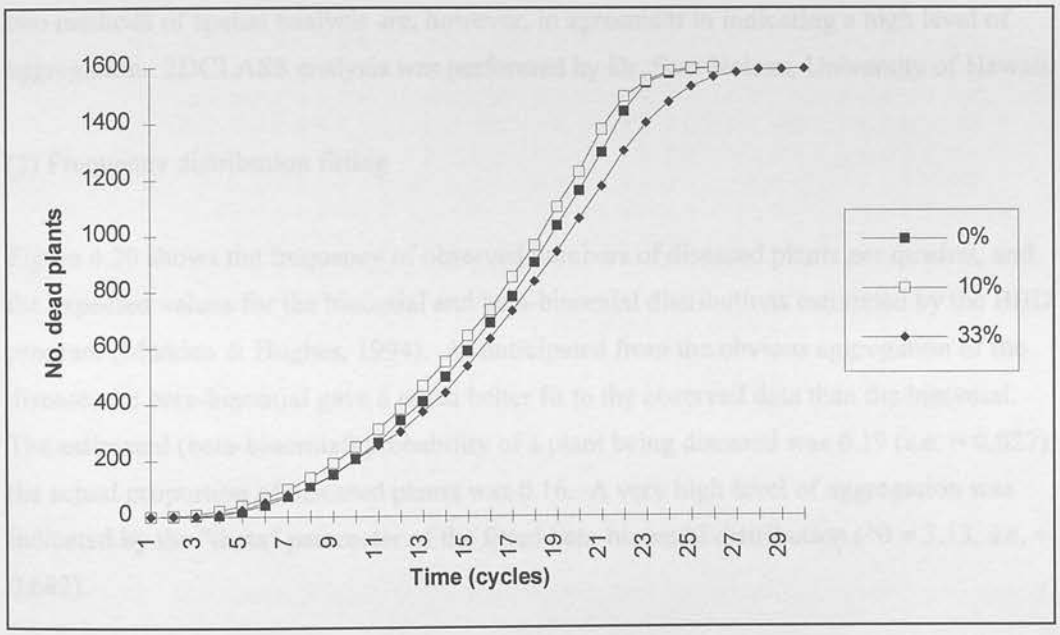
Field = 40 X 40 = 1600 plants; primary infection focus at Y=10, X=10; temperature = 12.0°C, σ = 0; rainfall is typical; prevailing wind from the North-West, with no variation in direction.

Figure 4.16d. The effect of inspecific infection failures on a simulated air-borne epidemic.



Field = 40 X 40 = 1600 plants; primary infection focus at Y=10, X=10; prevailing wind from the North-West, with no variation in direction; wind speed = 3.0 m/s, σ =0; 100% infection success.

Figure 4.17a. The effect of temperature on a simulated soil-borne epidemic.



Field = 40 X 40 = 1600 plants; primary infection focus at Y=10, X=10; temperature = 12.0°C, σ .= 0; wind speed = 5.0m/s, σ = 0; prevailing wind from the North-West, with no variation in direction.

Figure 4.17b. The effect of inspecific infection failures on a simulated soil-borne epidemic.

One map from the early stage of one of the simulated epidemics was arbitrarily selected to illustrate the methodology for spatial pattern analysis. The map chosen for analysis (Figure 4.18) simulated an aggregated pattern of diseased plants. The primary disease focus at the centre of the map had generated one other major cluster of infected plants toward the North-Eastern corner, and at least one other minor cluster. Under the user-specified wind conditions of this realisation of the simulation, there was little spread of the disease into the lower half of the field.

(1) 2-Dimensional distance class analysis

The 2DCLASS analysis of the chosen map is shown in Figure 4.19. The aggregated pattern of diseased plants is indicated in the 2DCLASS output by the large number (1268 out of 1600, 79.25%) of distance classes with standardised count frequencies (SCFs) significantly greater than expected ($p <= 0.05$) or less than expected ($p >= 0.95$). Interestingly in this example the 2DCLASS analysis underestimated the number of clusters (only one is indicated on the 2DCLASS output) and overestimated the cluster size. The results of the two methods of spatial analysis are, however, in agreement in indicating a high level of aggregation. 2DCLASS analysis was performed by Dr. Scot Nelson, University of Hawaii.

(2) Frequency distribution fitting

Figure 4.20 shows the frequency of observed numbers of diseased plants per quadrat, and the expected values for the binomial and beta-binomial distributions estimated by the BBD program (Madden & Hughes, 1994). As anticipated from the obvious aggregation of the disease, the beta-binomial gave a much better fit to the observed data than the binomial. The estimated (beta-binomial) probability of a plant being diseased was 0.19 (s.e. = 0.027), the actual proportion of diseased plants was 0.16. A very high level of aggregation was indicated by the "theta" parameter of the fitted beta-binomial distribution ($\hat{\theta} = 3.13$, s.e. = 0.662).

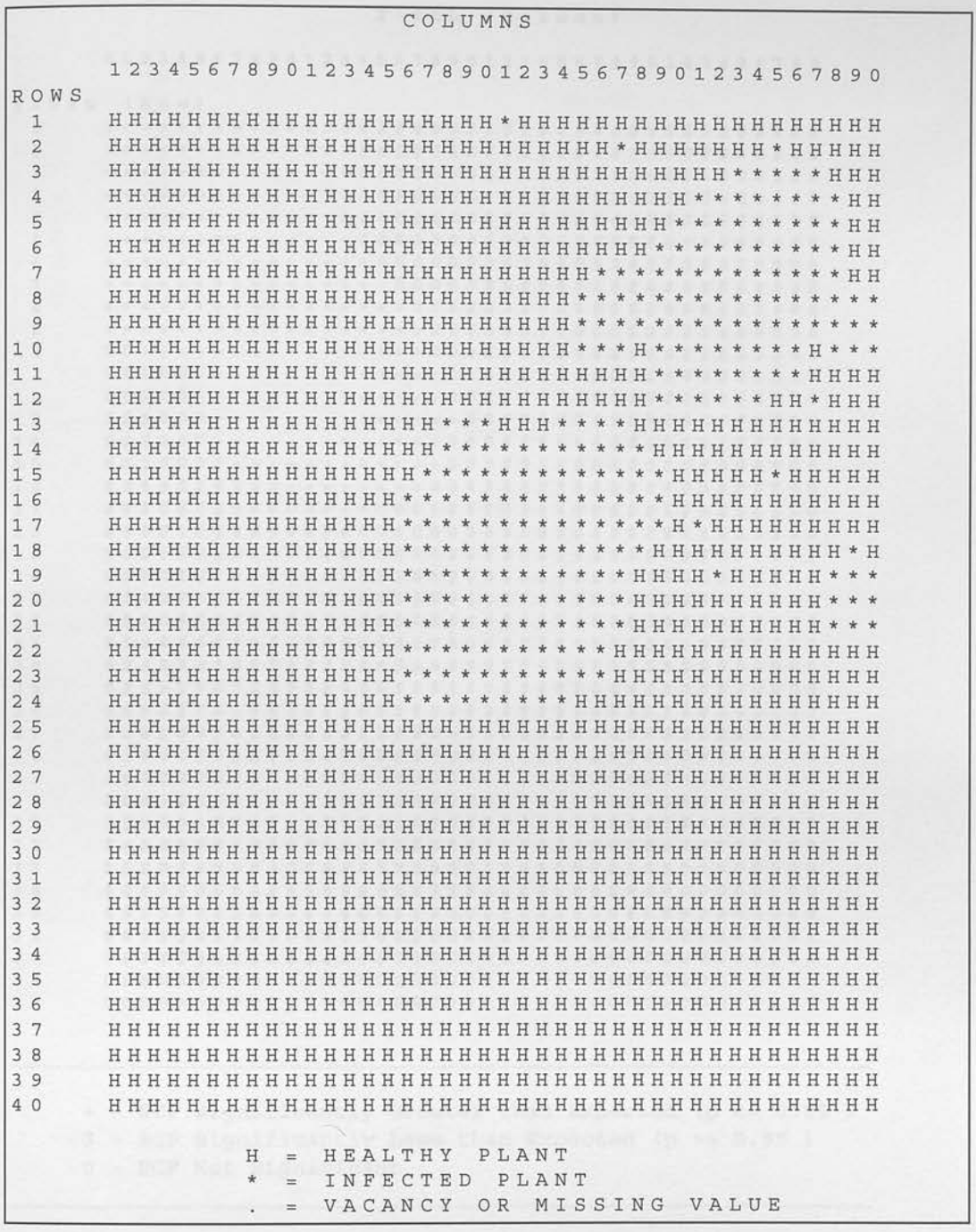


Figure 4.18. Map from an early stage of a simulated air-borne epidemic, used subsequently for spatial pattern analysis.

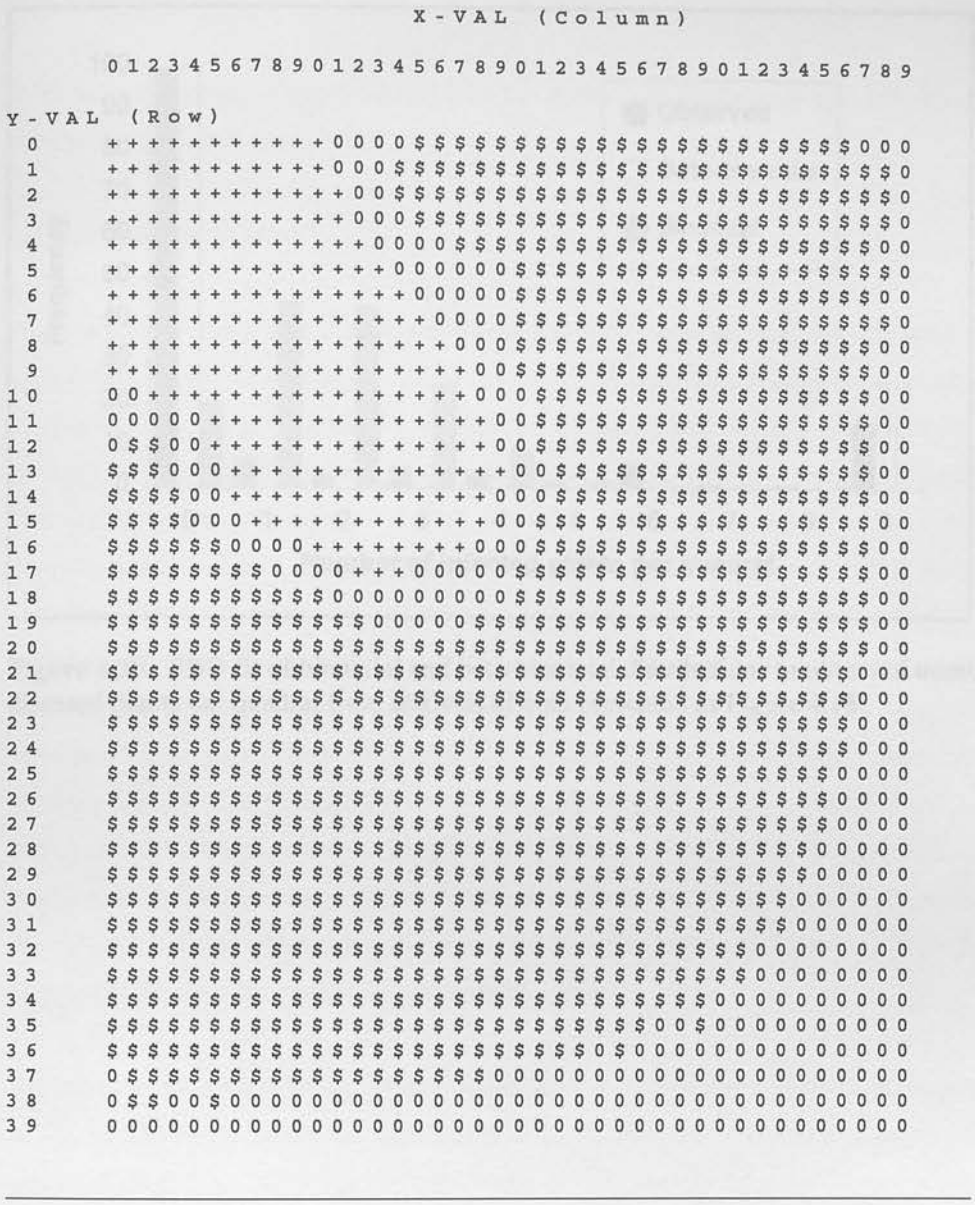


Figure 4.19. 2DCLASS analysis of SATSUMA ouput presented as Figure 4.18. (Analysis conducted and results provided by Dr. Scot Nelson, University of Hawaii).

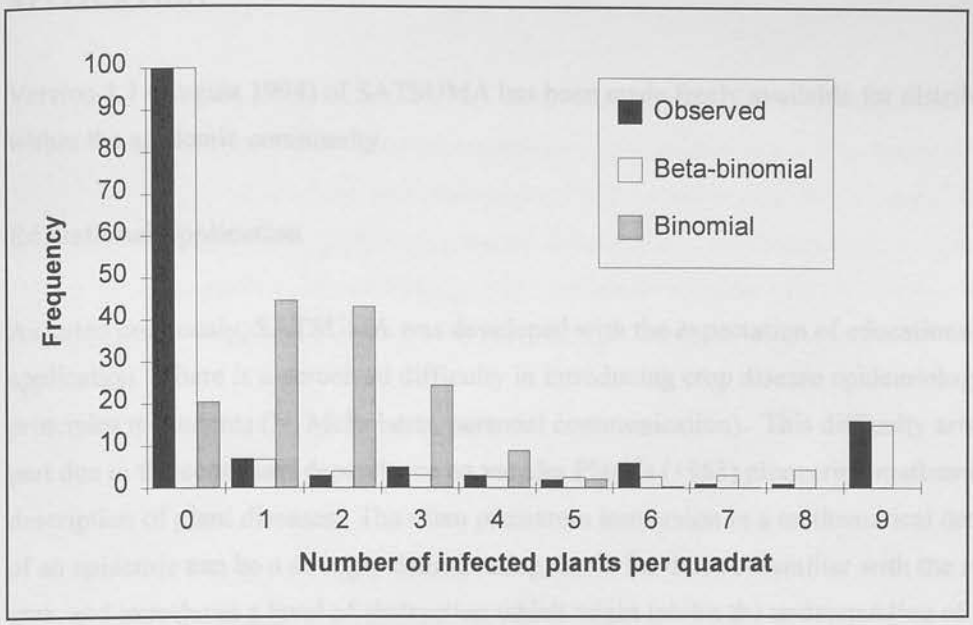


Figure 4.20. BBD fit of binomial and beta-binomial distributions to observed numbers of diseased plants per quadrat from analysis of map presented as Figure 4.18.

APPLICATION

Version 1.1 (August 1994) of SATSUMA has been made freely available for distribution within the academic community.

Educational application

As noted previously, SATSUMA was developed with the expectation of educational application. There is a perceived difficulty in introducing crop disease epidemiology principles to students (N. McRoberts, personal communication). This difficulty arises in part due to the continued dependence on van der Plank's (1963) pioneering mathematical description of plant diseases. The often premature immersion in a mathematical description of an epidemic can be a strongly demotivating factor for those unfamiliar with the subject area, and introduces a level of abstraction which might inhibit the understanding of underlying biological processes and patterns. This is apparently the case for the LATEBLIGHT model, which was developed from observationally-derived data, and which simulates the temporal development of a late blight epidemic through the course of a growing season. The development of the epidemic is presented as a growth curve, and the model therefore takes as its starting point a familiarity on the part of the user with the mathematical functions (e.g. logistic function) commonly used to describe epidemics. In this respect, what is otherwise a well designed and developed model, is unsuitable for use as an introductory model of crop disease epidemiology.

SATSUMA, by presenting a pictorial evolution of the epidemic, allows for an appreciation of the mathematical (temporal) description of the epidemic to develop in parallel with an understanding of its visual (spatial) development. By presenting a spatial development of the epidemic, a highly visual, essentially qualitative representation of disease progress, model users obtain a reference point (which might best be described as a "feel" for the data) to which the temporal description of the epidemic, output to file, can be related. A prototype version of the model was trialed by undergraduate horticulture students (N. McRoberts, unpublished results) and was received favourably, to the extent that course material is undergoing revision to include a laboratory exercise based on SATSUMA.

Research application

Despite having been developed primarily as an educational model of a mechanistic nature, with no internal reference to observationally-derived empirical functions, demonstrated correlation between simulated epidemic patterns and accepted empirical descriptions are good. SATSUMA has, to date, been adopted as a research tool in two areas in which a simulation approach confers advantage over traditional experimentation, specifically in the early stages of research as a means of gathering data quickly and cheaply.

Evaluation of disease sampling protocols

In order to compare the relative efficiency of disease sampling protocols, it is necessary to know the "true" incidence of disease, and the extent of patchiness, in the experimental system. This requirement introduces very real problems for the evaluation of sampling protocols using a real crop/disease interaction, since it is almost impossible to produce an exactly known incidence of disease in a crop. One approach which can circumvent this problem is to use a real crop, but to simulate the disease (Seem *et al.*, 1985). The approach taken by Seem *et al.* (1985) was to generate an artificial disease epidemic by placing a known number of plastic disks on the leaves of grape vines to simulate infection by *Plasmopara viticola.* The relative efficiency of different sampling protocols to detect incidence of simulated disease was assessed by their ability to detect the plastic disks placed within the vineyard. The downy mildew caused by *Plasmopara viticola* must be treated by preventative rather than curative fungicide treatments, and the primary motivation for sampling in this case is to detect the disease at as low an incidence as possible, since the pathogen is polycyclic and capable of very rapid increases in infection under suitable climatic conditions.

Field-scale experimentation clearly allows several factors in disease sampling to be examined (e.g. time of season, observer effects, and variation in cultivars) but it is expensive in comparison with the simulation approach. While protocols for disease detection can be assessed using SATSUMA, it is also possible to use the model to assess the relative merits of sampling protocols where an accurate assessment of disease incidence is required. In this type of sampling the aim is to obtain an estimate of the true disease incidence by taking a number of samples from the crop. Typically, diseased plants would be assumed to be either randomly distributed in the crop or to have an aggregated distribution. Based on an assumption about the true distribution of the disease this estimate allows disease management decisions to be made. As an additional consideration,

opportunity is presented at the end of an experiment to evaluate the efficiency of disease management actions.

A simple illustrative example of the use of SATSUMA for this type of assessment was conducted using the map listed as Figure 4.18. The field was divided initially into a grid of contiguous, 8*8 plant, square quadrats, the outside two rows of the map all round were disregarded. Plants were recorded as either diseased (infected or infectious, and dead) or healthy. In the first sampling procedure every quadrat in the grid was assessed. In subsequent procedures the initial 64 plant quadrats were divided into four 4*4 plant square quadrats and this grid was assessed by two methods. In the first the entire grid was again assessed, and in the second a "W" sample pattern (Southey, 1985) was followed across the map and the number of diseased plants in quadrats crossed by the W was noted. In the final method the 4*4 quadrats were divided into two rectangular 4*2 quadrats and a further W sample was conducted. The data were fitted to a beta-binomial distribution using the BBD program (Madden & Hughes, 1994) to obtain estimates of disease incidence (p) and degree of aggregation (θ). The true incidence of disease was calculated directly from the map. To compare the relative accuracy of the different sampling protocols in predicting the true incidence of diseased plants the deviation between the estimated and true incidence of diseased plants was calculated for each protocol and scaled by the number of plants assessed (Table 4.7).

Table 4.7. Assessment of sampling protocol accuracy conducted on a SATSUMA simulated population.

Sampling pattern	Plants per quadrat	θ	p	Estimated number of disease plants	Deviation from observed incidence	Scaled inaccuracy of sampling
Contiguous	64	1.02	0.13	186	-37	-2.62%
Contiguous	16	2.86	0.17	236	13	0.90%
W	16	3.26	0.20	272	49	3.54%
W	8	7.13	0.20	280	57	4.10%
Actual			0.16	223		

It can be seen that the estimated disease incidence (p) from the beta-binomial distribution was higher than the true incidence of disease for all bar one of the sampling protocols. The most accurate estimate of the true value of p was obtained from an assessment of the contiguous grid of 16 plant quadrats. In practice it would be impractical to assess disease incidence in a whole crop using a grid of contiguous quadrats since this involves sampling the whole crop, and the point of sampling to is to make an estimate of the true incidence

from a much smaller sample. There was no difference in the estimate of p provided by the two W sampling protocols and on this basis alone the method using the smaller 8 plant quadrats would probably be the more efficient since in a real situation sampling would be faster (and therefore cheaper) using the smaller quadrats.

In situations in which a particular disease incidence constitutes a threshold for treatment, the analysis outlined above can be extended to examine the efficiency of different sampling protocols to correctly identify situations in which treatment is needed. For any disease assessment system there are four possible outcomes as shown in the following Figure 4.21.

Outcome of sampling	True value of p below threshold	True value of p equal to or above threshold
estimate of p below threshold	**Decision: No treatment** *Correct*	**Decision: No treatment** *Incorrect*
estimate of p above threshold	**Decision: Treatment applied** *Incorrect*	**Decision: Treatment applied** *Correct*

Figure 4.21. Possible outcomes from sampling protocols intended to determine action in a threshold-based disease management system.

A simulation approach can be used to assess the efficiency of different sampling protocols in prompting the correct decision about treatment for different thresholds. The importance of a sampling protocol reaching the wrong decision (either to apply unnecessary treatment in the case when true p is below the threshold and estimated p is above it, or not to apply a treatment on the basis of an estimated p which is below the threshold in a case when true p is above it) can be determined by setting economic values for the cost of treatment, loss of yield when p is above its threshold and the cost of treatment.

For future research development, the demonstrated ability of SATSUMA to generate epidemics that conform to the beta-binomial distribution has great importance. The flexibility of the beta-binomial distribution means that it is not necessary to state *a priori* whether a disease has an aggregated or random spatial pattern, since the beta-binomial distribution collapses (automatically) to the binomial distribution in the absence of aggregation. SATSUMA can therefore be used to evaluate the efficiency of sampling protocols for diseases in which spatial distribution varies from near random to highly aggregated, and could therefore be used to develop sampling protocols which are adaptable to different degrees of disease aggregation. A research project of this type using SATSUMA has been planned for the near future at the University of Edinburgh (G. Hughes, personal communication).

Validation of spatio-temporal models

As noted in the introductory section spatio-temporal models of crop disease epidemics have been difficult to develop. A purely theoretical system for the description of spatio-temporal changes in disease incidence has recently been suggested (McRoberts & Hughes, 1994) in which the beta-binomial (spatial) and general logistic (temporal) functions are combined. Currently the model system uses disease incidence as an indication of the temporal progress of the epidemic, since disease incidence is directly related through the logistic function. Disease incidence in this context has previously been identified as "the biological time" of the epidemic (Kosman & Levy, 1994). In order to validate the new spatio-temporal model it will be necessary to parameterise it with reference to a true temporal scale. Suitable field data are not currently available and would be prohibitively expensive to collect for the initial validation procedure. The demonstrated qualitative agreement of SATSUMA output with real data will allow its use as a tool for the initial stages of validation of the spatio-temporal model.

DISCUSSION

Analysis

SATSUMA is a stochastic model with a number of randomly-specified variables, particularly in the routines simulating environmental conditions. A given set of user-defined variables yields a range of disease epidemics. Upon repetition, the user would expect to see an underlying, or "typical", pattern. The results of the temporal analyses reported here show that the model meets this expectation. It is also interesting to note that output from this mechanistic model shows broad qualitative agreement with general observations from empirical (equation-based) models to which stochastic parameters are applied (Renshaw, 1991). The derived values for the Gompertz function parameters showed an acceptable level of variation, while the generated disease progress curves always conformed to a generalised sigmoid shape. As an educational tool therefore, the model generates sufficient variation from a fixed set of input variables to encourage discussion of the forces which cause variation in the host*pathogen*environment interaction.

Currently the program does not include any procedures for directly analysing the disease progress curves generated by the simulation model. The user is forced to extract the relevant data from the tabular data output file and enter them into a suitable analysis program. While experienced plant pathologists may find this cumbersome, manipulating the data in this way generates a familiarity with the data which is not engendered if all analysis tasks are handled automatically. This notwithstanding, future development of the model should include an ability to view the disease progress curves from within the program.

As with the results of the temporal analysis the spatial analyses presented here can be verified by comparison with real data. For example, a similar 2DCLASS analysis to that shown here can be found in Ristaino *et al.* (1993). Handling of the spatial data by dividing the field into quadrats and collecting resulting disease incidence data introduces inexperienced users to the basic principles of these approaches. It should be possible in future developments of the model to include a procedure for producing the quadrat data for subsequent analysis. There is again, however, a balance to be struck between automating this task and the educational value of encouraging the user to obtain the data manually.

Geostatistical analyses were not performed on SATSUMA output. In very general terms, the method centres on a random pattern of core samples (in geology, bore or blast samples)

across the target area. Working under the assumption (generally valid in geological terms) that changes in the structure of the substrate (e.g. rock) will follow predictable gradients between sampling points (Webster & Oliver, 1990), interpolation among these samples generates a contour map of the distribution under consideration. The principal advantage of this analysis method is that spatial information is retained, and the visual output is directly comparable to the source map. The method provides detailed information about cluster size and intensity, and if the assumption of change predictability holds, the predictive quality of the method is good. Numerical data is generated along with the contour map, and hence provides potential for integration with temporal analysis techniques. Disadvantages are few but important. The smoothing techniques (an interpolation method referred to as "Kriging") forming an integral component of the method are computationally demanding, and computer software to conduct these computations is not generally available. A great many samples are required to produce the contour maps for output. A possible criticism of the methodology is that it was developed for use in geology, where, as already stated, rock structure follows predictable patterns. Particularly in the case of soil-borne pests of crops, there is evidence that this assumption is invalid (Boag & Neilson, 1994), rendering the entire analysis method inapplicable in this context.

Limitations and developmental considerations

Van der Plank (1975) noted several apparent errors in early computer simulation models of crop diseases, and cautioned plant pathologists against a whole-hearted acceptance of the value of simulation modelling. Certain assumptions have been incorporated into the model, which has been developed from concepts rather than from derived empirical relationships. SATSUMA is a stochastic model. Because there are a number of random variables, particularly in the routines simulating environmental conditions, a given set of user-defined parameters would be expected to yield a range of epidemic patterns. Upon repetition, the user would expect to see an underlying, or "typical", pattern. No such conclusions could be derived from a single run of the model, by virtue of the inherently random nature of many of the model's component routines. SATSUMA has been designed as a mechanistic, as opposed to deterministic model. While this undoubtedly produces a slightly simplistic output, this mechanistic approach has tangible benefit for developing an understanding of the basic mechanisms of fungal spread. This notwithstanding, testing of the model has confirmed that it produces simulated disease epidemics which conform to widely used empirical models of temporal and spatial disease development. With this conformity established, the stochastic nature of the model would have tangible benefit with respect to application-oriented research, for example the validation of disease monitoring protocols,

where a constant sampling methodology could be assessed against changing data obtained from a single, unchanging set of environmental parameters.

Genetic factors

In the development of SATSUMA, it was decided to discount genetic variation within both the crop and pathogen populations. The issue of genetic variability and its effects on a fungal epidemic is complex and a simulation challenge in its own right. Qualitative genetic factors act as simple YES/NO switches, and operate as a preliminary to all other factors. In the event that these genetic attributes, of both the pathogen and the host crop, are not suitable for proliferation of the fungus, all "downstream" factors (temperature etc.) become irrelevant. At a population level, genetic factors operating to the detriment of fungal proliferation would retard the rate of the epidemic. The rate of epidemic in a heterogeneous crop population is given by:

$$R_m = R_s + C[ln(m)]$$

where R_m is the rate in the heterogeneous host population, R_s is the rate in a homogeneous susceptible population (in effect the potential maximum rate), C is the generation interval measured in an equivalent unit to that used to express the rate of epidemic, and $ln(m)$ is the natural logarithm of the proportion of resistant genotypes in the crop population.

In the model presented, genetic variation within the crop population is ignored. The crop population is therefore taken to be genetically homogeneous, such as would be the case for a finished variety or pure-breeding line, or a clonal population. The assumption of genetic uniformity would not be valid in the case of a natural population, a fully or partially outbreeding crop, an unfinished breeding line, a land-race, varietal mixture or a multiline variety. With respect to the crop population, a mixture of genotypes would give rise to greater complexity in the spatial patterns of disease progress (Mundt *et al.*, 1986).

Susceptibility, or otherwise, to the pathogen is a feature of the host phenotype. Phenotypes could be assigned randomly in response to user-specified distribution probabilities. Each plant in the field would display a proportional susceptibility to the pathogen, which would operate "upstream" of temperature and other environmental variables in determining the success rate of attempted infections. This course of action would significantly increase the computational complexity of the model, with the field of defined phenotypes being stored as an array, and the probability of susceptibility to the pathogen of each and every plant being

assessed with each infection cycle. For a 20 plant by 20 plant field this would involved 400 mathematical operations per infection cycle, in place of the single assessment of host-susceptibility of the current model. The inclusion of this treatment of the crop population was deemed to yield insufficient benefit (given the original objective of developing a general educational tool) to justify the expected performance penalty.

There is an expectation of genetic variation within the pathogen population, and the phenomenon to be simulated would therefore be the interaction between host and pathogen genotypes. The fecundity of a polycyclic fungal organism would be expected to yield a high degree of mutation-induced genetic variation in consequence of the high number of reproductive cycles per unit time, and it can be assumed that this genotypic variation would cause a range of virulence levels within the pathogen population. In order to model this, it would be necessary to consider the relative reproductive contribution of each pathogen genotype (which we can safely assume would not be equal), and the consequent genetic constitution of the spore population released each infection cycle. An appreciation of the genetic interactions between host genotypes and pathogen genotypes would also be required, as pathogen virulence would depend in part upon pathogen-host mutual compatibility. For example, a highly susceptible target plant might escape infection by virtue of relatively ineffective spores, whereas a highly resistant host would fall victim because it has by chance been targeted by an extremely virulent pathogen genotype. The pattern of infection would give every impression that the first plant had a higher level of resistance to the pathogen than the second, a conclusion that is as incorrect as it is forgivable. It was decided that simulating pathogen virulence in this detail was inappropriate (in light of the approximations and generalisations already made), and settled for the specification of a single parameter to simulate host susceptibility/ pathogen virulence.

In SATSUMA, host susceptibility/pathogen virulence considerations have effect only during the infection phase of the pathogen life-cycle. While an unfavourable susceptibility/virulence combination will prevent the infection of a healthy target plant, it will in no way inhibit the maturation of the pathogen in an already infected plant, and future developments to SATSUMA will need to account not only for genetic influences on rate of infection, but also on the probability of post-infection fungal development.

Environmental factors

Different pathogen-host species combinations would be expected to show characteristically different curves for infection probability with respect to temperature. While our temperature function is acceptable for a general teaching model, the simulation of specific fungal pathogens would require a more accurate consideration of temperature and its influence on the epidemic. An alternative, and more sophisticated, temperature function, would define an asymmetrical curve, biased towards higher temperatures.

$$y_i = c_1\left(\frac{\exp\left[c_2\left(T_i - c_4\right)\right] - \exp\left[c_2\left(c_5 - c_4\right) - \left(c_5 - T_i\right)\right]}{c_3}\right)$$

where y_i denotes the temperature incremented from $i = 1$ to 25, $c_1 = 11.2$, $c_2 = 0.3225$, $c_3 = 100$, $c_4 = 1$ and $c_5 = 25$. The epidemic profile under these conditions describes a Logan curve (Logan *et al.*, 1976), with maximal pathogen spread at approximately 22°C.

Wind direction obviously has a rate-determining effect on the temporal and spatial development of the pathogen population. While the current version of SATSUMA utilises wind direction to determine the spatial pattern of the spread from both primary and secondary infection foci, the model is unrealistic in that the initial placement of secondary infection foci is not determined by wind direction. The extent to which a potential target plant is at risk of infection depends upon its location with respect to an infectious neighbour (which might be expressed as "degree of downwindness"), and would be dependent in turn upon the proximity of the target plant to the infectious individual. This is especially relevant to the spread of air-borne (and to a lesser extent splash-dispersed) pathogens, where released spores would be expected to spread downwind from the point of release, and show a greater incidence of secondary infection in areas closer to the point of release.

Two simple mathematical models are considered:

The Power Law model gives the number of infections formed, λ, as:

$$\lambda = \frac{c_1}{D^{c_2}}$$

where D is the distance from the source, and c_1 and c_2 are constants.

The exponential model gives the number of infections, λ, as:

$$\lambda = y^{(-cD)}$$

where y is the amount of disease at source, or the amount of spore deposited on the target, D is the distance from source, and c is a constant.

The mechanistic nature of the simulation, and especially its representation of the field as a two-dimensional grid of discrete entities, does not lend itself to the incorporation of the essentially continuous distribution models above. Whereas these formulae are ideal when considering distances in a continuous measurement, such as millimetres or miles, they are quite inappropriate for application to discontinuous, or "digital", units, such as the numbers-of-plants used in SATSUMA. SATSUMA therefore places secondary infections within the field in a uniformly random manner. While this approximation clearly leads to a reduction in descriptive and predictive accuracy of the simulation, it was felt that the retention of simplicity was of greater importance. As an additional consideration, the distance relationships described relate to a single source of infection, with all variables being defined relative to that source. In the simulated field, with the exception of the initial infection cycle from any given focus, there would not be a single infectious source. Rather, infectious plants form a pattern, or "front" of infection, which does not lend itself easily to the calculation of distance of a potential target plant from an infectious source. This, combined with the observation that in most instances the epidemic will be spreading simultaneously from more than one source, encouraged the acceptance of the simplified simulation of the infection process.

In the case of polycyclic splash-dispersed pathogens, the treatment of rainfall and its effect on the rate and pattern of disease spread, involves a greater degree of approximation than is desirable. In the context of a simple teaching model, it is wished only to illustrate the strong correlation between rainfall and dispersal, and the reduced influence of wind speed in this class of pathogen as compared to air-borne pathogens. It is commonly assumed that splash-dispersed pathogens are dispersed over shorter distances than air-borne pathogens. The distance of spore dispersal would be determined, amongst other things, by the velocity and angle at which the raindrop strikes the spore-bearing structure, and the characteristics of its deflection from the fungus and/or host plant surface. The range and direction of dispersal would then be influenced by wind speed and direction. In the model, the assumption of correlation between these factors and the amount of rainfall during the pathogen life-cycle is

made, and the maximum range of infection in response to the level of rainfall is specified. The deposition gradients for splash-dispersed spores are generally much steeper than those for air-borne pathogens. The decline in spore concentration is generally approximated by the exponential model, described earlier. Although the effects of wind speed are unclear in relation to splash-dispersal, there are observations that indicate that splash-dispersed spores are carried further in moving air than in still air (Fitt *et al.*, 1989). Field observations on a number of pathogens have shown that wind direction does have an influence on the dispersal of splash dispersed pathogens. Given the symbolic representation of the crop population, the mechanistic as opposed to purely mathematical simulation of infection and spread, and the stated aims of the model, it was felt that the approximations are defensible in terms of the simplicity of the model and the clarity of the output. Future developments will require a more detailed consideration of the role of rainfall in fungal epidemics.

Particularly in the case of soil-borne pathogens, cultivation practices would be expected to have significant effect on the pattern of pathogen dispersal. As an illustrative example, pathogens of crops cultivated in rows, such as grapes, raspberries and potatoes, would be expected to spread more rapidly along the rows, with cross-infection between rows being a relatively infrequent event. One might also expect rapid pathogen dispersal along vehicle tracks, and any feature of human activity that would result in the dispersal of pathogen-bearing soil. Future developments to the model will consider deviation from a uniform pattern of dispersal.

Pathogen life-cycle

The differentiation of the fungal life-cycle into distinct, non-overlapping reproductive and maturation phases, while perhaps acceptable at an organism level, is unrealistic at the population level. There is no theoretical justification for the maintenance of synchrony among pathogen generations. Just as genotypic variation within the pathogen population would give rise to phenotypic variation for virulence, so a phenotypic variation in generation interval among the evolving pathotypes might be expected. The pathogen life-cycle was divided into distinct infectious and maturation phases to prevent, in the case of favourable environmental conditions, the immediate transition of a newly infected plant to the infectious condition, and subsequently promote the infection of its neighbours. This would obscure the underlying pattern of the epidemic, and for this reason the division of the pathogen life-cycle is a justifiable approximation. The same argument can be used to defend the active maintenance of synchrony among generations.

The current version of SATSUMA assumes a unidirectional progression among the four host plant conditions, with all plants having the potential to become infected, and no probability of recovery from the infected state. This is intuitively incorrect, as witnessed by the phenomenon of genetically-conferred resistance to fungal infection and subsequent development. The ability of a plant to stabilise and survive infection is as important to plant breeders as the property of resistance to the initial infection. The model would be improved by the incorporation of a routine to confer, subject to conditions, the ability to confine infections and maintain growth. Equally, the assumption that an infected plant must pass through each stage of infection, without omission, is questionable. I am unaware of any theory preventing an infected plant from being immediately infectious. Despite this, in the case of fungal pathogens it is usually (almost exclusively) observed that infectious plants constitute a subset of the infected plants, and that usually (again, almost exclusively) plants become infectious only after a latent period when they are already infected. However, with air/splash-dispersed bacterial pathogens such as *Xanthomonas campestris* (the cause of black rot in cabbage) it is possible for the disease to progress with no discernible latent period, and plants can become infectious immediately upon becoming infected (Ruissen & Kocks, 1993). Future improvements to the model would ideally include the simulation of development processes within the host plant population.

In summary, SATSUMA represents a novel contribution to the field of simulation modelling in plant disease epidemiology by simulating both the spatial and temporal development of epidemics. Despite the early words of Shrum (1975) and the near universal recognition of his contribution to simulation modelling in this field, prior to the development of SATSUMA no fully implemented disease simulator had considered both the spatial and temporal aspects of epidemics. The only example of regard to spatial considerations prior to the development of SATSUMA involved the spatial distribution of host genotypes. Adaptations of the EPIMUL simulator (Kampermeijer & Zadoks, 1977) were developed which allowed the user to specify the spatial arrangement of different host genotypes and to specify whether disease is initiated at a number of randomly assigned points or in specific foci. After these initial specifications are completed, however, the adapted EPIMUL simulator presents the developing epidemic only as a disease progress curve and further spatial components are not considered apart from the effect on the disease progress curve of the previously specified arrangement of the host genotypes (Mundt & Leonard, 1986; Mundt *et al.*, 1986).

The mechanistic operation of SATSUMA, with each simulated plant being respresented by a spatially unique entity, is ideally suited to the spatial modelling of plant disease

epidemics. If a purely mathematical treatment of the host population is considered, the spatial consequences of infection from an individual infectious source cannot be modelled. Progression of infection by the action of set mechanisms upon an infectious source not only allows for study of the spatial evolution of the simulated epidemic, but also generates the data from which a temporal description of the epidemic can be generated.

The potential contribution of SATSUMA can best be appreciated by a comparison with the EPIDEMIC model introduced by Shrum (1975). EPIDEMIC's modular approach allowed a series of disease-specific offspring models to be developed. The procedures which determine the different components of pathogen activity in the current implementation of SATSUMA contain purely mechanistic functions. This notwithstanding, since each of these procedures is functionally independent, they can be replaced by compatible procedures in which pathogen activity is coded by empirically derived functions in an analogous manner to EPIDEMIC. SATSUMA can therefore be seen as a template for a novel approach to the simulation of the spatial and temporal dynamics of specific crop/disease interactions.

V. General Summary

In the development and subsequent use, or misuse, of models, one must firstly evaluate the system to be modelled, in order to ascertain whether or not the study of the model confers advantage over the direct study of the modelled system. Secondly, the validity of any model produced must be questioned. Is the modelled system sufficiently well understood to allow for the construction of an accurate model? What degree of inaccuracy is tolerable, or even desired? Questions of this nature have to be addressed and answered by the designers and users of the model, and the objectives and limitations of the model clearly stated, and understood by users of the model.

A simulation model can be used to describe the behaviour of a system, and subject to satisfactory validation to predict the behaviour of that system. Experimentation with the model can generate data where experimentation with the system is not possible, not practical or, of increasing influence, not affordable. It must constantly be borne in mind, however, that any model is no more than a representation of reality. Study of the model will not, and cannot, replace study of the modelled system. The definition of "simulate" uses words such as "feign", "pretend" and "mimic" (O.E.D., 1982). Whether good or bad, a model remains no more than an imitation of the system of interest, but where the system cannot be directly studied, a model is preferable to ignorance. A good model, when used correctly, will provide a valuable complement to the established scientific activities of observation, description and experimentation.

A modelling methodology was adopted and consistently implemented in the development of the models described here. Computer programs were coded in a common programming language, Pascal. A modular approach was adopted throughout, with logically discrete mechanisms, or groups of mechanisms, coded into separate, autonomous program procedures. This structured methodology resulted in models that have a potential beyond the realisation of the immediate goals of the specific modelling exercise. By designing computer programs whose physical structure approximates the functional structure of the modelled system, the simulation models produced are more readily accessible to scrutiny and amendment by others. Emphasis is placed on the consideration and understanding of the system rather than on the consideration and understanding of the model. A model constructed in this way becomes suitable as a platform for onward development, refinement and customisation, and therefore achieves a value beyond that conferred by the realisation of the initial modelling objectives.

Two computer-based simulation models have been presented, both are similar and yet very different. Both models feature a strongly mechanistic approach to the modelling of

composite phenomena, simulating physical mechanisms of change rather than taking a "black box" approach through the use of descriptive mathematical functions and formulae. Both models were inspired by earlier, analogue simulation models, and have improved on their predecessors by the introduction of a stochastic approach. As stated previously, both models were coded in the same programming language, on the same computer, in the same structured, procedure-based style.

PSELECT (Partner *et al.*, 1993) simulates population response to applied directional selection. This system had been modelled previously (Simmonds, 1979; Open University, 1987). The objective of this modelling exercise was to implement a new approach to modelling the system, to improve on the original design through the use of a more structured modelling methodology, to enable the addition of features (e.g. alternative breeding protocols and allelic dominance relationships) unable to be incorporated in either of its predecessors. As an additional feature, the model allows for automatic repetition of simulation runs, generating data that can be described by variability as well as value. Model output from PSELECT was successfully validated against analogue simulation data, and qualitative conformity to experimental selection responses was demonstrated. PSELECT achieved its stated objectives of improved model design and features, but has yet to find application beyond that achieved by its predecessors. This observation may reflect on the subject area as much as on the model.

Through the development of the PSELECT model, valuable modelling skills and techniques have been learnt. These methods were applied to the simulation of plant disease epidemics, a subject area in which simulation modelling activity and interest are intense. The computer simulation model SATSUMA models component mechanisms of disease development, to graphically simulate the spatial and temporal development of disease epidemics in crop populations. SATSUMA structure and function were directly inspired by PSELECT, and the models share a number of common mechanisms. SATSUMA output was shown to be suitable for analysis by methods of temporal and spatial pattern analysis in current use in the field of epidemiology, and demonstrated both qualitative and quantitative agreement with accepted empirical descriptions of disease progress with respect to time. A major feature of SATSUMA is its ability to generate spatial pattern data, a significant contribution to the subject area. In contrast to the PSELECT model, SATSUMA has already aroused interest, has found use in education, and in addition has been provisionally adopted for research application in Great Britain and the U.S.A.

Both models achieved their stated objectives, namely the simulation of the chosen system through the modelling of component mechanisms. Both models feature stochasticity as an intrinsic feature, both have made new contributions to their respective fields of study. That is not to say that the models cannot be improved.

Future development of PSELECT should allow for the simulation of polyploidy, and unequal contribution of loci. The model would also be improved by the ability to vary selection parameters (e.g. selection pressure and population size) from generation to generation, thus duplicating more accurately conditions experience in real breeding programmes. With specific respect to the PSELECT model, unless the systems of inheritance of the modelled character become better understood, model validation remains a major obstacle to continued development. Further effort might be more profitably directed to the continued development of the SATSUMA model. Demonstrated demand for the model suggests that attention to the model-user interface might be advisable, if the model is to receive widespread acceptance and use. Simulated field size is currently restricted by system capabilities. Field size could be increased by modelling plant units at the pixel level, as opposed to using a symbolic character set to denote plant condition. The model currently ignores genetic factors within both the host and pathogen populations, and this is a highly tempting area for continued development activity. Plant mechanisms allowing for the stabilisation of infections could be incorporated. Finally, a major positive attribute of SATSUMA is its highly visual output of the spatial development of a disease epidemic. The model would benefit from the ability to generate a temporal (growth curve) description concurrently with the display of spatial information. These observations concur with the opinion expressed by Wagenet (1991) who stated, with reference to the likely outcome of a modelling exercise, that "*the light shines most clearly not on the answers, but on the next questions*".

Computer-based simulation models should be regarded as means to an end, and not as ends in themselves. The physical products of modelling activity (the models) are readily obvious. A principal, and often ignored benefit of the modelling exercise is the development not of the model, but of the modeller (Wagenet, 1991). In the development of a model, the modeller's knowledge of the subject system(s) and modelling expertise develop beyond their original levels. It is the increased worth of the modeller that will likely prove of more subsequent scientific benefit than the computer code that was originally the focus of the modelling programme. The original, perhaps naïve (but aesthetically pleasing), notion was to form a bridge between the disparate disciplines of agriculture and computer science. The component disciplines of this research programme are not in themselves new, but it is

believed that this particular synthesis of ideas and techniques is a new and worthwhile academic contribution. I was not the first to design and develop computer-based agricultural simulation models. I will not be the last. If this contribution proves of use or benefit to those that follow then it will have achieved its objectives.

REFERENCES

Akhtar M. Ali (Personal communication). Mr. Manzoor Ali Akhtar was a postgraduate researcher at The Edinburgh School of Agriculture, from 1990 to 1992.

Allchin F.R. (1969). Early cultivated plants in India and Pakistan. In P.J. Ucko & G.W. Dimbleby (eds.), The domestication and exploitation of plants and animals: 323-329. Duckworth, London.

Alstad D., Curtsinger J., Abrams P. & Tilman D. (1991). POPULUS Release 1.46. Department of Ecology, Evolution & Behavior, University of Minnesota, MN 55455-0302.

Anderson T.R. (1991). Computer Modelling of Agroforestry Systems. Ph.D. thesis, University of Edinburgh.

ANSI (1978). American National Standard for information systems programming language FORTRAN secretariat, Computer and Business Equipment Manufacturers Association. New York Association for Computing Machinery.

ANSI (1983). American National Standard Programming Language PASCAL, ANSI/IEEE 770 X3.97-1983, American National Standards Institute, New York.

Arneson P.A., Oren T.R., Loria R., Jenkins J.J., Goodman E.D. and Cooper W.E. (1979). APPLESCAB: A pest management game. *NACTA J.*, **23**: 61-62.

Atkinson A.C. & Pearce M.C. (1976). The computer generation of Beta, Gamma and Normal random variables. *J. Roy. Stat. Soc.*, **A139**: 431-448.

Aust H.J., Hau B. and Kranz J. (1983). EPIGRAM - A simulator of barley powdery mildew. *Z. Pflanzenkr. Pflanzenschutz*, **90**: 244-250.

Aylor D.E., McCartney H.A. & Bainbridge A. (1981). Deposition of particles liberated in gusts of wind. *J. Appl. Meteorol.*, **20**: 1212-1221.

Bailey L.H. (1930). The cultivated Brassicas. II. *Gentes Herb.*, **2**: 211-267.

Berger R.D. (1976). Computer simulation of *Cercospora apii* in mixed populations of susceptible and tolerant celery. *Phytopathology*, **63**: 535-537.

Berger R.D. (1981). Comparison of the Gompertz and logistic equations to describe plant disease progress. *Phytopathology*, **71**: 716-719.

Berry L. (Personal communication). Dr. Lorraine Berry was a postgraduate researcher at The Edinburgh School of Agriculture, from 1987 to 1992.

Birtwistle, G.M. (1973). Simula Begin. Studentlitteratur, Lund.

Blodgett F.M. (1941). A method for the determination of losses due to diseased or missing plants. *American Potato Journal*, **18**: 132-135.

Boag B. & Neilson R. (1994). Nematode aggregation and its effect on sampling strategies. *Aspects of Applied Biology*, **37**: 103-111.

Booker L.B., Goldberg D.E. & Holland J.H. (1987). Classifier Systems and Genetic Algorithms. University of Michigan Cognitive Science and Machine Intelligence Laboratory, Technical Report No. 8.

Borland International Inc. Scotts Valley, CA 95067-0001.

Box G.E.P. & Muller M.E. (1958). A note on the generation of random normal deviates. *Ann. Math. Stat.*, **29**: 610-611.

Bratley P., Fox B.L. & Schrage L.E. (1987). A Guide to Simulation, 2nd Edn. Springer-Verlag, New York.

Bruhn J.A., Bruck R.I., Fry W.E., Arneson P.A. and Keokosky E.V. (1980). LATEBLIGHT: a plant disease management game. User manual. Cornell Univ. Dept. Plant Path. Mimeo 80-1, Ithaca, NY 14853.

Bruhn J.A. & Fry W.E. (1981). Analysis of potato late blight epidemiology by simulation modeling. *Phytopathology*, **71**: 612-616.

Bujakiewicz P. & van den Bosch P.P.J. (1991). A structured language for modelling and simulation of mixed continuous and discrete-event systems. Unpublished manuscript.

Buri P. (1956). Gene frequency in small populations of mutant *Drosophila*. *Evolution*, **10**: 367-402.

Burkill I.H. (1930). The Chinese mustards in the Malay Penninsula. *Gads' Bull.*, **5**: 99-117.

Butt D.J. & Xu X-M. (1993). A computerised apple scab warning system for use on farms. *Understanding Plant Disease Epidemics*. BSPP Presidential Meeting, Southampton, 13th-16th December 1993.

Campbell C.L. & Madden L.V. (1990). Introduction to Plant Disease Epidemiology. John Wiley & Sons, New York.

Campbell C.L. & Noe J.P. (1985). The spatial analysis of soilborne pathogens and root diseases. *Ann. Rev. Phytopathol.*, **23**: 129-148.

Campbell C.L. & van der Gaag D.J. (1993). Temporal and spatial dynamics of microsclerotia of *Macrophomina phaseolina* in three fields in North Carolina over four to five years. *Phytopathology*, **83**: 1434-1440.

Carefoot G.L. & Sprott E.R. (1967). Famine on the Wind. Rand McNally, Chicago.

Chellemi D.O., Rohrbach K.G., Yost R.S. & Sonoda R.M. (1988). Analysis of the spatial pattern of plant pathogens and diseased plants using geostatistics. *Phytopathology*, **78**: 221-226.

Chellemi D.O. & Marois J.J. (1991). Development of a demographic growth model for *Uncinula necator* using a microcomputer spreadsheet. *Phytopathology*, **81**: 250-254.

Clocksin W.F. & Mellish C.S. (1981). Programming in Prolog. Springer-Verlag, Berlin.

Coakley S.M. (1988). Variation in climate and prediction of disease in plants. *Ann. Rev. Phytopathol.*, **26**: 163-181.

Cochran W.G. (1936). The statistical analysis of field counts of diseased plants. Supplement to *J. R. Statist. Soc.*, **3**: 49-67

Crosby J.L. (1961). Teaching genetics with an electronic computer. *Heredity*, **16**: 255-273.

Dahl O-J, Myhrhaug B. & Nygård U. (1967). The Simula 67 Common Base Language. Norwegian Computing Centre, Oslo.

Davies R. & O'Keefe R. (1989). Simulation Modelling with Pascal. Prentice-Hall, London.

Dean P.G. & Murphy P.J. (1973). An application of computer simulation to the teaching of genetics in the upper secondary school. *J. Biol. Edu.*, **7**: 25-30.

Demeke T., Kidane Y. and Wuhih E. (1979). Ergotism, a report on an epidemic, 1977-1978. *Ethiopian Med. J.*, **17**: 107-114.

Digital Equipment Corporation, Maynard, MA 01754-2571.

Egley G.H. & Duke S.O. (1985). Physiology of weed seed dormancy and germination. In: Duke S.O. (ed.) Weed Physiology Vol.1, Reproduction and Ecophysiology (pp. 27-64). CRC Press, Boca Raton.

Ellis B. & Stroustrup B. (1990). The Annotated C++ Reference Manual. Addison-Wesley, Reading.

Elston, D.A. (1989). An Introduction to Mathematical Modelling - Course notes. Scottish Agricultural Statistics Service, Edinburgh.

Falconer D.S. (personal communication). Professor D.S. Falconer, now retired, was Director of the AFRC Unit of Animal Genetics, 1968-1980, and head of the University of Edinburgh Department of Genetics, 1969-1977.

Falconer D.S. (1981). Introduction to Quantitative Genetics, 2nd edn. Longman, London.

Ferris H. (1976). Development of a computer simulation model for a plant-nematode system. *J. Nematol.*, **8**: 255-263.

Ferris H. (1978). Modification of a computer simulation model for a plant-nematode system. *J. Nematol.*, **10**: 198-201.

Fick G.N. (1975). Heritability of oil content in sunflowers. *Crop Sci.*, **15**: 77-78.

Fitt B.D.L., McCartney H.A. & Walklate P.J. (1989). The role of rain in dispersal of pathogen inoculum. *Ann. Rev. Phytopathol.*, **27**: 241-270.

Fitter A. (1978). An Atlas of the Wild Flowers of Britain and Northern Europe. Collins, London.

Frandsen K.J. (1943). The experimental formation of *Brassica juncea*. *Dansk Bot. Archiv.*, **11**: 1-17.

Fuller J.G. (1968). The Day of St.Anthony's Fire. Macmillan, New York.

Gillespie D. (1989). P2C, a Pascal to C Translator. 256-80 CalTech, Pasadena CA 91125.

Gilligan C.A. (1985). Construction of temporal models: III. disease progress of soil-borne pathogens. In: Gilligan C.A. (ed.) Advances in Plant Pathology, Vol. 3 (pp. 67-105). Academic Press, London.

Gottwald T.R, Richie S.M. & Campbell C.L. (1992). LOCR2 - Spatial Correlation Analysis Software for the Personal Computer. *Plant Dis.*, **76**: 213-215.

Gregory P.H. (1961). The Microbiology of the Atmosphere. Leonard Hill, London.

Gurevich B.I., Filippov A.V. and Tverskoi D.L. (1979). Forecasting the development of harmfulness of potato late blight (*Phytophthora infestans*) under different meteorological conditions on the basis of a simulation model "Epiphtora". *Mikol. Fitropathol.*, **13**: 309-314.

Haldane J.B.S. (1949) The rate of mutation of human genes. *Hereditas*, suppl. vol. 1949: 267-273.

Harberd D.J. (1972). A contribution to the cytotaxonomy of Brassica (Cruciferae) and its allies. *Bot. J. Linn. Soc.*, **65**: 1-23.

Hau B. (1985). Epidemiologische simulatoren als Instrumente der Systemanalyse mit besonderer Berucksichtigung eines Modells der Gerstenmehltaus. *Acta. Phytomed.*, **9**: 1-101.

Hau B. (1988). Modelling epidemics of polycyclic foliar diseases and development of simulators. In Kranz J. & Rotem J. (eds.) Experimental Techniques in Plant Disease Epidemiology (pp. 267-277), Springer-Verlag, Berlin.

Hau B. (1990). Analytic models of plant disease in a changing environment. *Ann. Rev. Phytopathol.*, **28**: 221-245.

Hau B., Eisensmith S.P. & Kranz J. (1985) Construction of temporal models: II. Simulation of aerial epidemics In: Gilligan C.A. (ed.) Advances in Plant Pathology, Vol. 3 (pp. 31-65). Academic Press, London.

Hemmingway J.S. (1976). Mustards. In N.W. Simmonds (ed.), Evolution of crop plants: 56-59. Longman, London.

Horsfall J.G. (1972). Genetic Vulnerability of Major Crops. National Academy of Science, Washington DC.

Hughes G. (personal communication). Dr. Gareth Hughes is a lecturer in Crop Science, Institute of Ecology & Resource Management, University of Edinburgh.

Hughes G. & Madden L.V. (1992). Aggregation and the incidence of disease. *Plant Path.*, **41**: 657-660.

Hughes G. & Madden L.V. (1993). Using the beta-binomial distribution to describe aggregated patterns of disease incidence. *Phytopathology*, **83**: 759-763.

Hughes G. & Nelson S.C. (1994). Spatial pattern analysis of plant virus diseases and its application. In: Madden L.V., Raccah B. & Tresh J.M. (eds.) Epidemiology and Management of Plant Virus Diseases (In press). Springer-Verlag, Berlin.

Hull P. (1977). An interactive computer program simulating selection for quantitative genetic characteristics. *J. Biol. Edu.*, **11**: 202-206.

Hull P. (1978). An interactive computer simulation program to construct adaptive landscapes and to simulate the changes expected with selection. *J. Biol. Edu.*, **12**: 21-26.

IBM Corporation. Armonk, NY 10504.

Jeger M.J. (1983). Analysing epidemics in time and space. *Plant Path.*, **32**: 5-11.

Jeger M.J. (1986). Asymptotic and threshold criteria in model plant disease epidemics. *Plant Path.*, **35**: 355-361.

Jeger M.J., Jones G.D. & Griffiths E. (1983). Disease spread of non-specialised fungal pathogens from innoculated point sources in intraspecific mixed stands of cereal cultivars. *Ann. Appl. Biol.*, **102**: 237-244.

Jensen K. & Wirth N. (1974). PASCAL User Manual and Report, 2nd Edition. Springer-Verlag, New York.

Kampermeijer P. & Zadoks J.C. (1977). A simulation of foci and epidemics in mixtures of resistant and susceptible plants. PUDOC, Wageningen.

Kernighan B.W. & Ritchie D.M. (1978). The C Programming Language. Prentice-Hall, New Jersey.

Kerr R., Pooley R.J. & Hillston J.A. (1990). Simula course notes. EASE Ltd., Edinburgh.

Kimura M. & Ohta T. (1971). Theoretical aspects of population genetics. Princeton University Press.

Kocks C.G.& Ruissen M.A. (1994). Spatial pattern analysis of black rot epidemics in cabbage using geostatistics. *Quantitative Ecology of Pests and Diseases*. Wageningen Agricultural University Ph.D. Summer School, Wageningen, 5th-8th September 1994.

Kosman E. & Levy Y. (1994). Fungal foliar plant pathogen epidemics: modelling and qualitative analysis. *Plant Path.*, **43**: 328-337.

Kowalski R. (1981). PROLOG as a programming language. Imperial College of Science and Technology, University of London. DoC 81/26.

Kranz J. (1979). Simulation of epidemics caused by *Venturia inaequalis* (Cooke) Aderh. *OEPP Bull.*, **9**: 235-242.

Kranz J., Mogk M. & Stumpf A. (1973). EPIVEN - Ein simulator für Apfelschorf. *Z. Pflanzenkr. Pflanzenschutz*, **80**: 181-187.

Large E.C. (1940). The Advance of the Fungi. Henry Holt, New York.

Law A.M. & Kelton W.D. (1991). Simulation modeling and analysis, 2nd edn. McGraw-Hill, New York.

Lewin R. (1993). California's lousy vintage. *New Scientist*, 17th April 1993: 27-31.

Lewontin R.C. (1974). The genetic basis of evolutionary change. Columbia University Press, New York.

Lippman S. (1989). A C++ Primer. Addison-Wesley, Reading.

Logan J.A., Wollkind D.J., Hoyt S.C. & Tanigoshi L.K. (1976). An analytic model for description of temperature dependent rate phenomena in arthropods. *Environ. Entomol.*, **5**: 1133-1140.

Madden L.V. & Hughes G. (1994). BBD - Computer software for fitting the beta-binomial distribution to disease incidence data. *Plant Dis.*, **78**: 536-540.

Marcotty M. & Ledgard H. (1987). The World of Programming Languages. Springer-Verlag, New York.

Marsaglia G. & Bray T.A. (1964). A convenient method for generating normal variables. *SIAM Review*, **6**: 260-264.

Massie L.B. (1973). Modeling and simulation of Southern corn leaf blight caused by race T of *Helminthosporium maydis* Nisik. & Mayake. Ph.D. thesis, Pennsylvania State University, State College, Pennsylvania.

Mathsoft Inc. Cambridge, MA 02139.

McCartney H.A. & Fitt B.D.L. (1985). Construction of dispersal models. In: Gilligan C.A. (ed.) Advances in Plant Pathology, Vol. 3 (pp. 107-143). Academic Press, London.

McRoberts, N. (Personal communication). Dr. Neil McRoberts is a lecturer in Plant Pathology, The Scottish Agricultural College, Auchincruive.

McRoberts N. & Hughes G. (1994). A protocol for the examination of the effect of disease aggregation on the rate of disease progress. *Quantitative Ecology of Pests and Diseases*. Wageningen Agricultural University Ph.D. Summer School, Wageningen, 5th-8th September 1994.4.

McRoberts N. & Partner P.L.R. (1994). A mechanistic simulation of the action of fungal pathogens, used as an introduction to the underlying mathematical description of crop disease epidemics. In: France J. (ed.), Proceedings of the 25th Meeting of the Agricultural Research Modellers' Group. *J. Ag. Sci.*, **123**: 154.

Mercado A.C. & Lantican R.M. (1961). The susceptibility of cytoplasmic male-sterile lines of corn to *Helmintosporium maydis* Nisikado and Miyake. *Philipp. Agric.*, **45**: 235-243.

Microsoft Corporation. Redmond, WA 98052-6399.

Minitab Inc. State College, PA 16801-2756

Mizushima U. (1950). Karyogenetic studies of species and genus hybrids in the tribe Brassiceae of Cruciferae. *Tohoku J. Agr. Res.*, **1**: 1-14.

Mizushima U. & Tsunoda, S. (1967). A plant exploration in Brassica and allied genera. *Tohoku J. Agr. Res.*, **17**: 249-276

Morinaga T. (1933). Interspecific hybridisation in Brassica. V. The cytology of F_1 hybrid of *B. carinata* and *B. alloglabra*. *Japan. J. Bot.*, **6**: 467-475.

Morinaga T. (1934). Interspecific hybridisation in Brassica. VI. The cytology of F_1 hybrids of *B. juncea* and *B. nigra*. *Cytologia*, **6**: 62-67.

Muetzelfeldt, R. (1991). Modelling the modelling process. *Aspects of Applied Biology*, **26**: 89-100.

Mundt C.C. & Leonard K.J. (1986). Analysis of factors affecting disease increase and spread in mixtures of immune and susceptible plants in computer-simulated epidemics. *Phytopathology*, **76**: 832-840.

Mundt C.C., Leonard K.J., Thal W.M. & Fulton J.H. (1986). Computerized simulation of crown rust epidemics in mixtures of immune and susceptible oat plants with different genotype unit areas and spatial distributions of initial disease. *Phytopathology*, **76**: 590-598.

Murphy P.J. (1983). An exercise in biometrical genetics based on a computer simulation. *J.Biol.Edu.*, **17**: 343-348.

Murphy P.J. (1984). A CAL-based distance education project in evolution: 1. Description of the project. *J. Biol. Edu.*, **18**: 37-44.

Murphy P.J. (1986). Computer simulations in biological education: analogues or models? *J. Biol. Edu.*, **20**: 201-205.

NAG Ltd. (1987). Genstat 5 reference manual. Clarendon Press, Oxford.

Naur P. (1963). Revised Report on the Algorithmic Language ALGOL 60. *Numerische Mathematik*, **4**: 420.

Neave H.R. (1973). On using the Box-Müller transformation with multiplicative congruential pseudo-random number generators. *Applied Statistics*, **22**: 92-97.

Nelson S.C., Marsh P.L. & Campbell C.L. (1992). 2DCLASS, a two-dimensional distance class analysis software for the personal computer. *Plant Dis.*, **76**: 427-432.

Nutter F.W. Jr., Teng P.S. & Shokes F.M. (1991). Disease assessment terms and concepts. *Plant Dis.*, **75**: 1187-1188.

O.E.D. (1982). The Concise Oxford Dictionary of Current English, 7th Edn. (ed. J.B Sykes). Oxford University Press.

Olsson G. (1960). Species crosses within the genus Brassica. I. Artificial *Brassica juncea* Coss. *Hereditas*, **46**: 351-396.

Open University (1987). Open University course material, S298 Genetics. Open University Press, Milton Keynes.

Padmanabhan S.Y. (1973). The great Bengal famine. *Ann. Rev. Phytopathol.*, **11**: 11-26.

Partner P.L.R., Smith M.L. & Spoor W. (1991). Crop pedigree selection: development of a simple simulation model. *Aspects of Applied Biology*, **26**: 281-284.

Partner P.L.R., Smith M.L., Spoor W. & Clarkson M.I. (1993). Computer simulation of selection in a hypothetical crop species. *CABIOS*, **9**: 597-605.

Pooley, R.J. (1987). An Introduction to Programming in Simula. Blackwell, Oxford.

Prakash S. & Hinata K. (1980). Taxonomy, cytogenetics and origin of crop Brassicas, a review. *Opera Bot.*, **55**: 1-57.

Press W.H., Flannery B.P., Teukolsky S.A & Vetterling W.T. (1989). Numerical recipes in Pascal - the art of scientific computing. Cambridge University Press.

Pritsker A.A.B. (1974). The GASP IV Simulation Language. John Wiley & Sons, New York.

Pritsker A.A.B. (1986). Introduction to Simulation and SLAM II. John Wiley & Sons, New York.

Pritsker A.A.B. & Krivat P.J. (1969). Simulation with GASP II: A Fortran Based Simulation Language. Prentice-Hall, New Jersey.

Rakow G. & Woods D.L. (1987). Outcrossing in rape and mustard under Saskatchewan prairie conditions. *Canadian Journal of Plant Science*, **67**: 147-151.

Rapilly F. (1977). Recherche des facteurs de resistance horizontal à la Septoriose du Ble (*Septoria nodorum* Berk.). Resultats Obtenus par la simulation. *Ann. Phytopathol.*, **9**: 1-9.

Rapilly F. (1979). Simulation d'une epidemie de *Septoria nodorum* Berk. sur ble, etude des possibilites de resistance horizontale. *EPPO Bull.*, **9**: 243-250.

Rapilly F. & Jolivet E. (1976). Construction d'un modele (EPISEPT) permettant la simulation d'une epidemie de *Septoria nodorum* Berk. sur ble. *Rev. Stat. Appl.*, **24**: 31-60.

Renshaw E. (1991). Modelling Biological Populations in Space and Time. University Press, Cambridge.

Rich S.S., Bell A.E. & Wilson S.P. (1979). Genetic drift in small populations of *Tribolium*. *Evolution*, **33**: 579-584.

Richards F.J. (1959). A flexible growth function for empirical use. *J. Exp. Bot.*, **10**: 290-300.

Rijsdijk F.H. (1975). A simulator of yellow rust of wheat. C. R. Semaine d'etude agriculture et hygiene des plantes. *Bull. Rech. Agron. Gemblous*: 411-418.

Ristaino J.B., Larkin R.P. & Campbell C.L. (1993). Spatial and temporal dynamics of *Phytophthora* epidemics in commercial Bell Pepper fields. *Phytopathology*, **83**: 1312-1320.

Roberts D.F., Billewicz W.Z. & McGregor I.A. (1978). Heritability of stature in a West African population. *Ann. Hum. Genet.*, **42**: 15-24.

Robertson A (1960). A theory of limits in artificial selection. *Proc. R. Soc. London* **B**, **153**: 234-249.

Rossiter D.G. (1991). Modern Computer Programming Techniques for Environmental Simulation Modelling. SCAS Teaching Series No.1. Cornell University, Ithaca, NY 14853.

Rouse D.I. (1985). Construction of temporal models: I. Disease progress of air-borne pathogens. In: Gilligan C.A. (ed.) Advances in Plant Pathology, Vol. 3 (pp. 11-29). Academic Press, London.

Rouse D.I. (1988). Use of crop growth-models to predict the effects of disease. *Ann. Rev. Phytopathol.*, **26**: 183-201.

Ruissen M.A. & Kocks C.G. (1993). Cabbage black rot: the epidemic development of a bacterial disease. *Understanding Plant Disease Epidemics*. BSPP Presidential Meeting, Southampton, 13th-16th December 1993.

Sall M.A. (1980a). Epidemiology of grape powdery mildew: a model. *Phytopathology*, 70: 338-342.

Sall M.A. (1980b). Uses of stochastic simulation: grape powdery mildew (*Uncinula necator*) example. *Z. Pflantzenschutz.*, **87**: 397-403.

Sasaoka T. (1930). Karyological observations in different interspecific hybrids of Brassica. *Japan. J. Genet.*, **6**: 20-32.

Schlager G. & Dickie M.M. (1967). Spontaneous mutations and mutation rates in the house mouse. *Genetics*, **57**: 319-330.

Seem R.C., Magarey P.A., McCloud P.I. & Wachtel M.F. (1985). A sampling procedure to detect grapevine downy mildew. *Phytopathology*, **75**: 1252-1257.

Shaner G.E., Peart R.M., Nenman J.E., Stirm W.L. and Lower O.L. (1972). EPIMAY, a plant disease display model: An evaluation of the computer simulator for Southern corn leaf blight in Indiana. Purdue Univ. Agric. Exp. Stn. Res. Bull. 890.

Shannon R.E. (1975). Systems Simulation: the Art and the Science. Prentice-Hall, New Jersey.

Shearer B.L. & Zadoks J.C. (1974). The latent period of *Septoria nodorum* in wheat. 2. The effect of temperature and moisture under field conditions. *Neth. J. Plant Pathol.*, **80**: 48-60.

Shrum R. (1975). Simulation of wheat stripe rust (*Puccinia striiformis* West.) using EPIDEMIC, a flexible plant disease simulator. Penn. State Univ. Agric. Expt. Stn. Prog. Rep. 347.

Simmonds N.W. (1979). Principles of Crop Improvement. Longman, London.

Simula a.s. Postbox 335, Blindern, 0313 Oslo 3.

Sinskaia E.N. (1928). The oleiferous plants and root crops of the family Cruciferae. *Bull. Appl. Bot. Genet. Pl. Breed.*, **19**: 1-648.

Southey J.F. (1985). Laboratory methods for work with plant and soil nematodes. H.M.S.O., London.

Spence I. (1993). Dynamic array bounds in C. *Computing*, 18th February: 34-35.

Sprague G.F. & Schuler J.F. (1961) The frequencies of seed and seedling abnormalities in maize. *Genetics* **46**: 1713-1720.

Stephan S. & Gutsache V. (1980). Ein algorithmisches Modell zur Simulation der *Phytopthora*-Epidemie (SIMPHYT). *Arch. Phytopathol. Pflanzenschutz.*, **16**: 183-191.

Strandberg J. (1973). Spatial distribution of cabbage black rot and the estimation of diseased plant populations. *Phytopathology*, **63**: 998-1003.

Sun V.G. (1970). Breeding plants of Brassica. *J. Agr. Assoc. China*, **71**: 41-52.

Swanson M.R., Dudley J.W. & Carmer S.G. (1974). Simulated selection in autotetraploid populations. *Crop Sci.*, **14**: 625-636.

Taha H.A. (1976). Operations Research, an introduction, 2nd Edition. Collier Macmillan, London.

Technojock Software Inc. P.O.Box 820927, Houston, TX 77282-0927.

Teng P.S. (1985). A comparison of simulation approaches to epidemic modeling. *Ann. Rev. Phytopathol.*, **23**:351-379.

Teng P.S., Blackie M.J. and Close R.C. (1980). Simulation of the barley leaf rust epidemic: structure and validation of BARSIM-1. *Agric. Syst.*, **5**: 55-73.

Tsunoda S. & Nishi, S. (1968). Origin, differentiation and breeding of cultivated Brassica. *Proc. XII Int. Congr. Genet.*, **2**: 77-88.

Tucker A.B. (1986). Programming languages, 2nd edn. McGraw-Hill, New York.

U N. (1935). Genomic analysis in Brassica, with special reference to the experimental formation of *B. napus* and peculiar mode of fertilization. *Japan J. Bot.*, 7: 389-452.

Ullstrup A.J. (1970). History of Southern corn leaf blight. *Plant. Dis. Rep.*, **54**: 1100-1102.

van den Bosch F., Zadoks J.C. & Metz J.A.J. (1988). Focus expansion in plant disease. II: Realistic parameter-sparse models. *Phytopathology*, 78: 59-64.

van der Plank J.E. (1963). Plant Diseases: Epidemics and Control. Academic Press, New York.

van der Plank J.E. (1975). Principles of Plant Infection. Academic Press, New York.

Vaughan J.G. (1956). The seed coat structure of *Brassica integrifolia* (West) O.E. Schulz var.*carinata* (A. Br.). *Phytomorph.*, **6**: 363-367.

Vaughan J.G., Hemmingway, J.S. and Schofield, H.J. (1963). Contributions to a study of variation in *Brassica juncea*. *J. Linn. Soc. (Bot.)*, **58**: 435-447.

Vavilov N.I. (1949). The origin, variation, immunity and breeding of cultivated plants. *Chron. Bot.*, **13**: 1-364.

Wagenet R.J. (1991). The future of modelling. *Aspects of Applied Biology*, **26**: 209-216.

Waggoner P.E. (1986) Disease progress curves. In: Leonard K.J. & Fry W.E. (eds.) Plant Disease Epidemiology, Vol. 1 (pp. 3-37). Macmillan, New York.

Waggoner P.E. & Horsfall J.G. (1969). EPIDEM, a simulator of plant disease written for computer. Conn. Agric. Exp. Stn. Bull. 698.

Waggoner P.E., Horsfall J.G. and Lukens R.J. (1972). EPIMAY, a simulator of Southern corn leaf blight. Conn. Agric. Exp. Stn. Bull. 729.

Waggoner P.E. & Parlane J.-Y. (1974a). Mathematical model for spore germination at changing temperature. *Phytopathology*, **64**: 605-610.

Waggoner P.E. & Parlane J.-Y. (1974b). Verification of a model of spore germination at variable, moderate temperatures. *Phytopathology*, **64**: 1192-1196.

Waggoner P.E. & Rich S. (1981). Lesion distribution, multiple infection, and the logistic increase of plant disease. *Proc. Natl. Acad. Sci. USA*, **78**: 3292-3295.

Webster R. & Oliver M.A. (1990). Statistical Methods in Soil and Land Resource Survey. Oxford University Press, Oxford.

Wheeler B.E.J. (1976). Diseases in Crops. Studies in Biology no. 64. Edward Arnold, London.

Williams P.H. & Hill C.B. (1986). Rapid-cycling populations of Brassica. *Science*, **232**: 1385-1389.

Wilson G. (1991). The accessibility of models - taking the side off the black box. *Aspects of Applied Biology*, **26**: 151-161.

Woodhouse-Smith C. (1962). The Great Hunger, Ireland 1845-1849. Harper & Row, New York.

Wright S. (1921). Systems of mating. *Genetics*, **6**: 111-178.

Wright S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proc. 6th Int. Congr. Genet.*, **1**: 356-366.

Wright S. (1951). The genetical structure of populations. *Ann. Eugen. Lond.*, **15**: 323-354.

Yardbird Software. Box 4646, West Hills, CA 91308.

Zadoks J.C. (1971). Systems analysis and the dynamic of epidemics. *Phytopathology*, **61**: 600-610.

Zadoks J.C. & Rijsdijk F.H. (1972). Epidemiology and forecasting of cereal rust, studied by means of a computer simulator named EPISIM. *Proc. Eur. Mediterr. Cereal Rust Conf.*, Prague **1**: 293-296.

Zadoks J.C. & Rijsdijk F.H. (1973). A calculated guess of future rust development in cereal crops. *INSA Bull.*, **46**: 440-443.

Zadoks J.C. & Schein R.D. (1988). James Edward Vanderplank: maverick and innovator. *Ann. Rev. Phytopathol.*, **26**: 31-36.

Zawolek M.W. (1993) Shaping a focus: wind and stochasticity. *Neth. J. Pl. Path.*, **99**, Supplement 3: 241-255.

Zeven A.C. & Zhukovsky, P.M. (1975). Dictionary of cultivated plants and their centres of diversity. PUDOC, Wageningen.

APPENDICES

APPENDIX A1

Anonymous contribution to the BCAB bulletin board

How to determine which programming language you're using
==========

The proliferation of modern programming languages which seem to have stolen countless features from each other sometimes makes it difficult to remember which language you're using. This guide is offered as a public service to help programmers in such dilemmas.

C: You shoot yourself in the foot.

C++: You accidentally create a dozen instances of yourself and shoot them all in the foot. Providing medical care is impossible since you can't tell which are bitwise copies and which are just pointing at others and saying "That's me, over there".

BASIC: Shoot self in foot with water pistol. On big systems, continue until entire lower body is waterlogged.

FORTRAN: You shoot yourself in each toe, iteratively, until you run out of toes, then you read in the next foot and repeat. If you run out of bullets, you continue anyway because you have no exception-processing ability.

PASCAL: You want to shoot yourself in the foot, but the gun has no firing pin and the compiler won't let you have one.

PROLOG: You attempt to shoot yourself in the foot but the bullet, failing to find its mark, backtracks into the gun which then explodes in your face.

ALGOL: You shoot yourself in the foot with a musket. The musket is aesthetically fascinating and the wound baffles the adolescent medic in the emergency room.

MODULA/2: After realising that you can't actually accomplish anything in the language, you shoot yourself in the head.

APL: You hear a gunshot and there's a hole in your foot, but you don't remember enough linear algebra to understand what happened.

(Anon., May 1994)

APPENDIX B1

Dictionary of variable names used in PSELECT

ARRAYS

Mean_of_repetitions: 2-dimensional ARRAY of REAL values, with dimensions Y=Selection_limit (Y<=9), X=Number of population values. Holds arithmetic mean values calculated across repetitions of the simulation.

Parental_phenotypes: vertical ARRAY of REAL values (Y=Number_selected, X=1) holding phenotypic scores from the Parents ARRAY.

Parents: ARRAY of selected plants, with INTEGER dimensions of maximum number of individuals selected (Y=100, allowing for a selection pressure of 0.5 to 0.005) by maximum number of loci (X=20).

Population: Central population ARRAY of INTEGER values, with dimensions of maximum population size (Y=100) by maximum number of loci (X=20).

Population_phenotypes: vertical ARRAY of REAL values (Y=Population_size, X=1) holding population phenotypic scores.

Results: 3-dimensional storage array of REAL values, dimensions of Z=Number of repetitions (Z<=50), Y=Selection_limit (Y<=9), X=Number of population values.

sd_of_repetitions: 2-dimensional ARRAY of REAL values, with dimensions Y=Selection_limit (Y<=9), X=Number of population values. Holds standard deviation values calculated across repetitions of the simulation.

sem_of_repetitions: 2-dimensional ARRAY of REAL values, with dimensions Y=Selection_limit (Y<=9), X=Number of population values. Holds standard error values calculated across repetitions of the simulation.

GLOBAL VARIABLES

Aggr_output: TEXT variable used as the logical (device) name for the output file to which aggregated results are written.

Allele_segregation: REAL value (0 to 1) defining the probability of segregation at heterozygous (A1A2) loci.

Breeders: INTEGER specifying the number of selected plants contributing to generation Ft+1 (defines the Y-dimension of the Parents ARRAY).

Continue_program: single character STRING variable accepting a user-prompt to continue program execution following an interruption.

Corrected_selection_pressure: REAL value correcting for rounding errors arising from the "ROUND(Num_selected)" statement (from the application of a REAL selection pressure to INTEGER plant units).

Counter: INTEGER variable used in CHARACTER STRING acceptance routines.

Data_output: TEXT variable used as the logical (device) name for the data output file to which results are written.

Dbug_output: TEXT variable used as a logical (device) name for a storage file used for monitoring/testing during debugging. No function in operational mode.

Degrees_of_freedom: INTEGER specifying the degrees of freedom from aggregated resuls (number of repetitions - 1).

Environment: REAL value, enviromental random deviate added to genotype to generate phenotypic score.

Environmental_sd: REAL value denoting standard deviation applied in the generation of normally-distributed Environment random deviates.

Exit_program: single character STRING variable accepting a user-prompt to exit the program at the completion of execution.

Female: INTEGER giving the identity of a designated female parent (in "Parents" ARRAY) in outcrossing.

Gen: INTEGER holding variable for Generation.

Gene_action: 1 to 9 character STRING variable denoting the nature of allelic dominance interactions at loci.

Generation: INTEGER value acting as a counter to denote the generation number, F_x.

Genotype: INTEGER value, the genotypic value of an individual plant.

Genotype_background: INTEGER value of constant genotypic component (genomic background score).

H2: REAL value, realised broad sense heritability. Calculated from the ratio of Var_genotype to Var_phenotype (Vg/Vp).

Heterozygosity: REAL value, level of population heterozygosity. Calculated arithmetically directly from Population ARRAY.

Heterozygote_value: REAL value denoting the value of the heterozygote (0.51 to 0.99) relative to the value of the selectively advantageous homozygote. Used in conjunction with **A1A2** constant to specify varying degrees of allelic dominance.

Heterozygous: INTEGER denoting the number of heterozygous loci in the population.

Incidence_of_A1_allele: INTEGER denoting the calculated occurrence of the A_1 allele within the Population ARRAY.

Incidence_of_A2_allele: INTEGER denoting the calculated occurrence of the A_2 allele within the Population ARRAY.

Locus: INTEGER variable (1 to 20) referencing the individual locus in the plant.

Male: INTEGER giving the identity of a designated male parent (in "Parents" ARRAY) in outcrossing.

Max_selection_pressure: REAL value calculated as 1-(1/Population_size), ensuring the selection of at least one plant from each generation.

Mean_environment: REAL value, the mean of population environmental values.

Mean_genotype: REAL value, the mean of population genotypic values.

Mean_phenotype: REAL value, the mean of population phenotypic values.

Migrants: INTEGER specifying the number of migrants entering the population per generation.

Migration: BOOLEAN variable, value TRUE enables Migrations procedure.

Migration_frequency: REAL variable defining mean migration frequency per generation.

Mutation: BOOLEAN variable, value TRUE enables Mutations procedure.

Mutation_U: REAL variable defining probability of forward mutation.

Mutation_V: REAL variable defining probability of back mutation.

Normal_method: 1 to 10 character STRING variable specifying which uniform to normal distribution transformation method is used.

Num_of_loci: INTEGER values 1 to 20, specifying the number of loci defining character.

Num_of_repetitions: INTEGER values 1 to 50, defining the number of experimental repetitions. The maximum potential value was iteratively determined as 91, but the variable is limited to <=50 for the benefit of system resources.

Number_selected: INTEGER values 1 to Population_size specifying the number of individuals selected each generation.

Offspring: INTEGER; reference to individual offspring from the contributing parental group.

Offspring_contribution: INTEGER value defining the number of offspring contributed to the next generation by each selected parent.

p: REAL value denoting the population allelic frequency of the A_1 allele.

Parent: INTEGER reference to an individual plant within the group of selected plants ("Parents" ARRAY).

Phenotype: REAL value, the phenotypic value of an individual plant.

Plant: INTEGER variable (1 to 200) referencing the individual plant within the population.("Population" ARRAY).

Pollination_mode: 1 to 4 character STRING variable specifying breeding protocol (self vs. monogamous or polygamous outbreeding).

Population_size: INTEGER values 1 to 200, defining population size per generation.

Population_value: INTEGER referencing the physical position of a population parameter within Results ARRAY.

q: REAL value denoting the population allelic frequency of the A_2 allele.

Quit_program: 1 to 3 character STRING variable accepting a user-prompt to abort program execution.

Repetition: INTEGER value uniquely identifying each experimental repeat.

RInt100: Random INTEGER between 1 and 100, inclusive.

RReal1: holding variable for random REAL number between 0.001 and 1.000.

Selection_basis: 1 to 3 character STRING variable specifying whether selection is defined by number of plants per generation, or by a stated selection pressure.

Selection_direction: 1 to 4 character STRING variable, with values UP or DOWN defining the direction of applied phenotypic selection.

Selection_limit: INTEGER values 1 to 9, giving the number of generations over which selection is practiced.

Selection_pressure: REAL value, 0 to Max_selection_pressure (appx.=1) specifying proportion of each genration failing to contribute gametes to the following generation (functionally lethal w.r.t. selection).

Sqrt_of_num_of_reps: REAL value (analogous to $\sqrt{n}$) used to calculate standard deviation and standard error of the mean across repetitions of the simulation.

ssq_environment: REAL value, the sum of squares of environmental values, used in the calculation of population statistics.

ssq_genotype: REAL value, the sum of squares of genotypic values, used in the calculation of population statistics.

ssq_of_repetitions: REAL value used in the calculation of standard deviation and standard error of the mean across repetitions.

Population_size: INTEGER values 1 to 200, defining population size per generation.

Population_value: INTEGER referencing the physical position of a population parameter within Results ARRAY.

q: REAL value denoting the population allelic frequency of the A_2 allele.

Quit_program: 1 to 3 character STRING variable accepting a user-prompt to abort program execution.

Repetition: INTEGER value uniquely identifying each experimental repeat.

RInt100: Random INTEGER between 1 and 100, inclusive.

RReal1: holding variable for random REAL number between 0.001 and 1.000.

Selection_basis: 1 to 3 character STRING variable specifying whether selection is defined by number of plants per generation, or by a stated selection pressure.

Selection_direction: 1 to 4 character STRING variable, with values UP or DOWN defining the direction of applied phenotypic selection.

Selection_limit: INTEGER values 1 to 9, giving the number of generations over which selection is practiced.

Selection_pressure: REAL value, 0 to Max_selection_pressure (appx.=1) specifying proportion of each genration failing to contribute gametes to the following generation (functionally lethal w.r.t. selection).

Sqrt_of_num_of_reps: REAL value (analogous to $\sqrt{n}$) used to calculate standard deviation and standard error of the mean across repetitions of the simulation.

ssq_environment: REAL value, the sum of squares of environmental values, used in the calculation of population statistics.

ssq_genotype: REAL value, the sum of squares of genotypic values, used in the calculation of population statistics.

ssq_of_repetitions: REAL value used in the calculation of standard deviation and standard error of the mean across repetitions.

ssq_phenotype: REAL value, the sum of squares of phenotypic values, used in the calculation of population statistics.

Sum_environment: REAL value, the calculated sum of environmental values, used in the calculation of population statistics.

Sum_genotype: REAL value, the calculated sum of genotypic values, used in the calculation of population statistics.

Sum_of_repetitions: REAL value derived from the arithmetic mean across repetitions of the simulation.

Sum_phenotype: REAL value, the calculated sum of phenotypic values, used in the calculation of population statistics.

TempInt: temporary holding pointer for INTEGER values.

TempReal: temporary holding pointer for REAL values.

Var_environment: REAL value, the variance of population genotypic values.

Var_genotype: REAL value, the variance of population genotypic values.

Var_phenotype: REAL value, the variance of population genotypic values.

LOCALLY-DECLARED VARIABLES

Answer: locally-declared STRING (or CHARACTER) variable. User response to screen prompt.

Answer_as_real: locally-declared REAL variable, used in the conversion of STRING response to REAL number.

Converted_OK: locally-declared INTEGER variable (operates as BOOLEAN) to confirm conversion of STRING value to REAL number.

Environmental_mean_correction: locally-declared REAL variable used in modified Box-Muller transformation.

i: counter INTEGER used in transformation of uniform deviates to normal distribution by Central Limits Theorem.

n: counter INTEGER specifying extent of sampling in transformation of uniform deviates to normal distribution by Central Limits Theorem.

Sdev_modifier: locally-declared REAL variable used in modified Box-Muller transformation.

Selection_parameters_approved: 1 to 3 character STRING variable, functioning as a BOOLEAN to accept user-specified selection parameter values into the model.

Sum_RReal1: REAL variable holding the cumulative total of uniform random deviates. Equates to the T variable in Central Limit Theorem transformation routine.

Summary_approved: 1 to 3 character locally-declared STRING variable. Returned values other than Y or YES prevent program progression, and repeat preceding routines.

T: REAL variable used in Central Limit Theorem algebraic procedure.

U1: locally-declared random REAL deviate, value 0.001 to 1, used in Marsaglia-Bray transformation routine.

U2: locally-declared random REAL deviate, value 0.001 to 1, used in Marsaglia-Bray transformation routine.

U3: locally-declared random REAL deviate, value 0.001 to 1, used in Marsaglia-Bray transformation routine.

V1: locally-declared random REAL deviate, value -1 to 1, used in Marsaglia-Bray transformation routine.

V2: locally-declared random REAL deviate, value -1 to 1, used in Marsaglia-Bray transformation routine.

FUNCTIONS

Accept_integer: FUNCTION operating on a locally specified question to verify and accept a user-specified INTEGER value into the model.

Accept_real: FUNCTION operating on a locally specified question to verify (and where necessary convert an INTEGER response to REAL) and accept a user-specified REAL value into the model.

CONSTANTS

A1A1: CONSTANT value 3, genotypic contribution of homozygous A_1 locus.

A1A2: CONSTANT value 2, genotypic contribution of heterozygous locus. Interactively odified within code to compensate for full or overdominance.

A2A2: CONSTANT value 1, genotypic contribution of homozygous A_2 locus.

Debug_on: CONSTANT value FALSE. Set to TRUE in program code if embedded debugging procedures are to be utilised.

Num_of_population_values: CONSTANT value 14, used in Population ARRAY to define number of columns in output file.

APPENDIX B2

PSELECT Program code

```
PROGRAM PSELECT;
USES Crt, DOS;

{ Comment:  Metric character selection model
            ------------------------------------------------
            Program: PSELECT
            ------------------------------------------------
            Prototype release 1.0b - 16/02/93
            Developed from:
            Turbo Pascal, production version SELECT06.PAS
            ------------------------------------------------
            Coded by Paul Partner
            October 1991 - February 1993                    }

{ Comment: Declaration of global constant values }

CONST
  A1A1=3;               { genotypic contribution of homozygous A1 locus }
  A1A2=2;               { genotypic contribution of heterozygous locus }
                        { - modified within code to compensate for full
                          dominance }
  A2A2=1;               { genotypic contribution of homozygous A2 locus }

  num_of_population_values=14;  { used in population array to define columns }

{ Comment: both "genotype_background" and "number_selected" can be
           constant, "static" variables, but unlike number_selected,
           genotype_background has to be calculated, and must therefore
           be declared as a dynamic variable, using the VAR statement.
           It would be better were number_selected also to be specified
           as such, but it could not then be used to define the upper
           limit of an array.

           Copy of statements, from VARIABLE procedure:

           1. genotype_background:=ROUND(((A1A1+A2A2)/2)*num_of_loci);
           2. number_selected:=ROUND(population_size*selection_pressure); }


{ Comment: Declaration of global dynamic variable values }

VAR

{ Comment: Declaration of dynamic variables that, aesthetically, should
           be declared as constants, but due to their derivation, must
           be declared as dynamic variables. }

  num_of_loci:INTEGER;              { number of loci defining character }
  population_size:INTEGER;          { population size per generation }
  number_selected:INTEGER;          { individuals selected each generation }
  selection_limit:INTEGER;          { number of generations of selection }
  migrants: INTEGER;                { mean number of migrants per generation }
  num_of_repetitions:INTEGER;       { number of experimental repetitions }

  genotype_background:INTEGER;      { value of constant genotypic component }
  offspring_contribution:INTEGER;   { number of offspring generated by
                                      each parent }

{ Comment: Declaration of dynamic variable REAL values }

  genotype:REAL;                { genotypic value }
```

```
    sum_genotype:REAL;          { sum of genotypic values }
    ssq_genotype:REAL;          { sum of squares of genotypic values }
    mean_genotype:REAL;         { mean of genotypic values }
    var_genotype:REAL;          { variance of genotypic values }
    phenotype:REAL;             { phenotypic value }
    sum_phenotype:REAL;         { sum of phenotypic values }
    ssq_phenotype:REAL;         { sum of squares of phenotypic values }
    mean_phenotype:REAL;        { mean of phenotypic values }
    var_phenotype:REAL;         { variance of phenotypic values }
    environment:REAL;           { enviromental factor added to genotype }
    environmental_sd:REAL;      { applied environmantal standard deviation }
    sum_environment:REAL;       { sum of environmental factors }
    mean_environment:REAL;      { mean of environmental factors }
    ssq_environment:REAL;       { sum of squares of environment values }
    var_environment:REAL;       { the realised environmental variance }
    allele_segregation:REAL;    { probability of A1A2 segregation }
    H2:REAL;                    { realised broad sense heritability, Vg/Vp }
    heterozygosity:REAL;        { population heterozygosity }
    selection_pressure:REAL;    { proportion of each genration failing to
                                  contribute to the following generation }
    Max_selection_pressure: REAL; { 1-(1/population_size) }
    corrected_selection_pressure: REAL; { correction for ROUND(num_selected }
    heterozygote_value:REAL;    { relative value of heterozygote, 0.51 to 0.99}
    Migration_frequency: REAL; { migrants as proportion of poulation size }
    sum_of_repetitions:REAL;    { used to calculate mean across repetitions }
    sqrt_of_num_of_reps:REAL;   { used to calculate s.d. and s.e.m across reps }
    ssq_of_repetitions:REAL;    { used to calculate s.d. and s.e.m across reps }
    p:REAL;
    q:REAL;

    TempReal:REAL;              { Temporary pointer to real value }
    RReal1:REAL;                { Random real number between 0.001 and 1.0 }

  { Comment: Declaration of dynamic variable INTEGER values }

    counter:INTEGER;            { used in CHARACTER STRING acceptance routines }
    locus:INTEGER;              { reference to individual locus in plant }
    plant:INTEGER;              { reference to individual plant within
                                  population ("POPULATION" array) }
    generation:INTEGER;         { the generation number, Fx }
    gen:INTEGER;
    repetition:INTEGER;         { identification of each experimental repeat }
    breeders:INTEGER;           { number of selected plants contributing to
                                  generation Ft+1 }
    outcross:INTEGER;           { generation where out-X_ing initiated }
    parent:INTEGER;             { reference to individual plant within group
                                  of selected plants ("PARENTS" array) }
    male:INTEGER;               { designated male parent in outcrossing }
    female:INTEGER;             { designated female parent in outcrossing }
    offspring:INTEGER;          { reference to individual offspring from
                                  parental contribution group }
    heterozygous:INTEGER;       { number of heterozygous loci in population }

    population_value:INTEGER; { reference to position of population parameter }
    degrees_of_freedom:INTEGER;  { number of repetitions, minus 1 }
    Incidence_of_A1_allele:INTEGER;
    Incidence_of_A2_allele:INTEGER;

    T:INTEGER;
    TempInt:INTEGER;            { Temporary pointer to integer value }
    RInt100:INTEGER;            { Random integer between 1 and 100, inclusive }

    Gene_action:STRING[9];        { Nature of allelic interaction at loci }
    normal_method:STRING[10];     { Which uniform -> normal method is used }
    Pollination_mode:STRING[4];   { Self vs. cross pollination }
    Selection_basis:STRING[3];    { Selection by no. plants or selection press }
    Selection_direction:STRING[4];
    Quit_program:STRING[3];       { User prompt to abort program }
    Continue_program:STRING[1];   { User prompt to continue program after
```

```
                                    interupt }
  Exit_program:STRING[1];         { User prompt to exit program after
                                    execution }

  Debug_On: BOOLEAN;
  Mutation: BOOLEAN;
  Migration: BOOLEAN;

{ Comment: Declaration of variable TEXT values }

  Data_output:TEXT;             { Output file for results }
  Aggr_output:TEXT;             { Output file for aggregated results }
  Dbug_output:TEXT;             { Output file for monitoring program execution }

{ Comment: Declaration of variable ARRAY values, both REAL and INTEGER }

  population:ARRAY [1..200,1..20] of INTEGER;
                  { 1..Population_size,1..num_of_loci }
  parents:ARRAY [1..100,1..20] of INTEGER;
               { 1..number_selected,1..num_of_loci }
  population_phenotypes:ARRAY [1..200] of REAL;
                                { 1..population_size }
  parental_phenotypes:ARRAY [1..100] of REAL;
                              { 1..number_selected }
  results:ARRAY[1..50,1..9,1..num_of_population_values] of REAL;
             { 1..num_of_repetitions,1..selection_limit,1.. }
  mean_of_repetitions:ARRAY[1..9,1..num_of_population_values] of REAL;
                          { 1..selection_limit,1.. }
  sd_of_repetitions:ARRAY[1..9,1..num_of_population_values] of REAL;
                        { 1..selection_limit,1.. }
  sem_of_repetitions:ARRAY[1..9,1..num_of_population_values] of REAL;
                       {1..selection_limit,1.. }


{ Comment: All FUNCTIONS to be called in the program, to be
           declared and specified. }


FUNCTION Accept_integer (Question:STRING; minimum, maximum:INTEGER): INTEGER;
  VAR
    Answer:STRING;              { User response to screen prompt }
    Answer_as_real:REAL;        { Conversion of STRING response to REAL number }
    Converted_OK:INTEGER;       { Confirmation of conversion to REAL number }
  BEGIN
  REPEAT
    REPEAT
      WRITE(Question);
      READLN(Answer);
      VAL(Answer,Answer_as_real,Converted_OK);
    UNTIL (Converted_OK=0);
  UNTIL (Answer_as_real >= minimum)
    AND (Answer_as_real <= maximum)
    AND (Answer_as_real = INT(Answer_as_real));
  Accept_integer:=TRUNC(Answer_as_real);
  END;


FUNCTION Accept_real (Question:STRING; minimum, maximum:REAL): REAL;
  VAR
    Answer:STRING;              { User response to screen prompt }
    Answer_as_real:REAL;        { Conversion of STRING response to REAL number }
    Converted_OK:INTEGER;       { Confirmation of conversion to REAL number }
  BEGIN
  REPEAT
    REPEAT
      WRITE(Question);
      READLN(Answer);
      VAL(Answer,Answer_as_real,Converted_OK);
    UNTIL (Converted_OK=0);
```

```
  UNTIL (Answer_as_real >= minimum)
    AND (Answer_as_real <= maximum);
  Accept_real:=Answer_as_real;
  END;

{ Comment: All PROCEDURES to be called in the program, to be
           declared and specified. }

PROCEDURE Open_DBug_Output;
  BEGIN
  ASSIGN(DBug_output,'PSELECT.BUG');
  REWRITE(DBug_output);
  WRITELN(DBug_output,'File: PSELECT.BUG.');
  WRITELN(DBug_output,'Program PSELECT.PAS execution progression record');
  WRITELN(DBug_output,' ');
  END;
{ Comment: Procedure "Open_DBug_Output" opens progress output file and
           echos procedure execution pointers to output. File is then
           closed pending subsequent APPEND statements }

PROCEDURE Close_DBug_Output;
  BEGIN
  CLOSE(DBug_output);
  END;
{ Comment: Procedure "Close_DBug_Output" concludes the procedural output
           and closes the output file }

PROCEDURE Correct_for_dominance;
  BEGIN
  IF Gene_action='DOM' THEN    { Correction for full allelic dominance }
    BEGIN
    IF population[plant,locus]=A1A2 THEN
      BEGIN
      genotype:=genotype+(A1A1-A1A2);
      END;
    END;
  IF Gene_action='PAR' THEN    {Correction for partial dominance }
    BEGIN
    IF population[plant,locus]=A1A2 THEN
      BEGIN
      genotype:=genotype+((A1A1-A2A2)*(heterozygote_value-0.5));
      END;
    END;
  END;

PROCEDURE Calculate_allelic_frequencies;
  BEGIN
  Incidence_of_A1_allele:=0;
  Incidence_of_A2_allele:=0;
  FOR plant:=1 to population_size DO
    BEGIN
    FOR locus:=1 to num_of_loci DO
      BEGIN
      CASE population[plant,locus] OF
      A1A1: Incidence_of_A1_allele:=Incidence_of_A1_allele+2;
      A1A2: BEGIN
            Incidence_of_A1_allele:=Incidence_of_A1_allele+1;
            Incidence_of_A2_allele:=Incidence_of_A2_allele+1;
            END;
      A2A2: Incidence_of_A2_allele:=Incidence_of_A2_allele+2;
      END;
      END;
    END;
  p:=Incidence_of_A1_allele/(Incidence_of_A1_allele+Incidence_of_A2_allele);
```

```
  q:=Incidence_of_A2_allele/(Incidence_of_A1_allele+Incidence_of_A2_allele);
  END;


PROCEDURE Ask_Population_size;
  BEGIN
  WRITELN;
  WRITELN
  ('Enter population size as an integer. The maximum permissible value is 200.');
  WRITELN
  ('Population size, once specified, remains constant among generations.');
  WRITELN;
  Population_size:=Accept_integer('Population size = ',1,200);
  WRITELN;
  CLRSCR;
  END;


PROCEDURE Ask_Number_of_loci;
  BEGIN
  WRITELN;
  WRITELN
  ('Enter number of loci defining character under selection. The value must');
  WRITELN
  ('be an integer not exceeding 20');
  WRITELN;
  num_of_loci:=Accept_integer('Number of loci = ',1,20);
  WRITELN;
  CLRSCR;
  END;


PROCEDURE Ask_Allelic_action;
  BEGIN
  WRITELN;
  WRITELN
  ('Specify nature of allelic interaction at each locus. The model currently');
  WRITELN
  ('does not support independent action at each locus. All loci will therefore');
  WRITELN
  ('be subject to the same mode of allelic interaction');
  WRITELN;
  WRITELN
  ('Enter ADD for additive dominance:');
  WRITELN
  ('          - The heterozygote value is exactly intermediate with respect');
  WRITELN
  ('            to the two homozygous forms.');
  WRITELN
  ('      DOM for full dominance:');
  WRITELN
  ('          - The heterozygote value is equal to that of the dominant');
  WRITELN
  ('            homozygote.');
  WRITELN
  ('      PAR for partial dominance:');
  WRITELN
  ('          - The heterozygote value exceeds the intermediate of the two');
  WRITELN
  ('            homozygotes, but is less than that of the dominant homozygote');
  WRITELN;
  REPEAT
    WRITE('Allelic action = ');
    READLN(Gene_action);
    FOR counter:=1 TO 9 DO Gene_action[counter]:=
                UPCASE(Gene_action[counter]);
  UNTIL (Gene_action='ADD')
     OR (Gene_action='DOM')
     OR (Gene_action='PAR');
```

```
  IF Gene_action='PAR' THEN
    BEGIN
    WRITELN;
    WRITELN
    ('Enter the relative value of the heterozygote, which must be between');
    WRITELN
    ('0.51 and 0.99.');
    WRITELN;
    heterozygote_value:=Accept_real('Value = ',0.51,0.99);
    END;
  WRITELN;
  CLRSCR;
  END;


PROCEDURE Ask_Breeding_regime;
  BEGIN

  IF Population_size=1 THEN
    BEGIN
    Pollination_mode:='SELF';
    WRITELN;
    WRITELN
    ('You have chosen a population size of 1, which precludes outcrossing.');
    WRITE
    ('Propagation will be by inbreeding. Press ENTER key to continue. ');
    READLN(Continue_program);
    WRITELN;
    END

  ELSE IF Population_size=2 THEN
    BEGIN
    WRITELN;
    WRITELN
    ('You have chosen a population size of 2, which precludes polygamy.');
    WRITELN;
    WRITELN('Specify breeding regime');
    WRITELN;
    WRITELN('Enter MONO for monogamous full outbreeding');
    WRITELN('      SELF for full inbreeding');
    WRITELN;
    REPEAT
      WRITE('Breeding regime = ');
      READLN(Pollination_mode);
      FOR counter:=1 TO 4 DO Pollination_mode[counter]:=
                  UPCASE(Pollination_mode[counter]);
    UNTIL (Pollination_mode='MONO')
       OR (Pollination_mode='SELF');
    WRITELN;
    END

  ELSE
    BEGIN
    WRITELN;
    WRITELN('Specify breeding regime');
    WRITELN;
    WRITELN('Enter MONO for monogamous full outbreeding');
    WRITELN('      POLY for polygamous full outbreeding');
    WRITELN('      SELF for full inbreeding');
    WRITELN;
    REPEAT
      WRITE('Breeding regime = ');
      READLN(Pollination_mode);
      FOR counter:=1 TO 4 DO Pollination_mode[counter]:=
                  UPCASE(Pollination_mode[counter]);
    UNTIL (Pollination_mode='MONO')
       OR (Pollination_mode='POLY')
       OR (Pollination_mode='SELF');
    WRITELN;
```

```
    END;

  CLRSCR;
  END;


PROCEDURE Ask_environmental_sd;
  BEGIN
  WRITELN;
  WRITELN
  ('Specify the applied environmental standard deviation. The greater the');
  WRITELN
  ('figure, the larger the environmental component of phenotypic variance.');
  WRITELN;
  WRITELN
  ('Figures between 0 and 15 would be reasonable.');
  WRITELN;
  environmental_sd:=
      Accept_real('Applied environmental standard deviation = ',0,100);
  WRITELN;
  CLRSCR;
  END;


PROCEDURE Ask_Transformation_method;
  BEGIN
  WRITELN;
  WRITELN
  ('Specify uniform to normal distribution transformation method');
  WRITELN;
  WRITELN
  ('The following statistical methods for generating a simulated environment');
  WRITELN
  ('are offered. For a full discussion, refer to Partner et al (1993).');
  WRITELN;
  WRITELN('Enter POLAR for Marsaglia-Bray Polar method');
  WRITELN('      BM for Box-Muller method');
  WRITELN('      CLT for Central Limit Theorem method');
  WRITELN;
  REPEAT
    WRITE('Transformation method = ');
    READLN(normal_method);
    FOR counter:=1 TO 10 DO normal_method[counter]:=
                  UPCASE(normal_method[counter]);
  UNTIL (normal_method='BM')
     OR (normal_method='POLAR')
     OR (normal_method='CLT');
  WRITELN;
  CLRSCR;
  END;


PROCEDURE Ask_Selection_basis;
  VAR
    Selection_parameters_approved:STRING[3];
  BEGIN
  WRITELN;
  REPEAT
    BEGIN
    CLRSCR;
    WRITELN
    ('Do you wish to select on the basis of:');
    WRITELN;
    WRITELN
    ('(a) Number of plants selected each generation? [Enter A]');
    WRITELN
    ('(b) Selection pressure? [Enter B]');
    WRITELN
    ('     (Selection pressure is defined here as being the proportion of');
```

```
    WRITELN
    ('     the population failing to contribute offspring to the subsequent');
    WRITELN
    ('     generation)');
    WRITELN;
    REPEAT
      WRITE
      ('Enter A or B ');
      READLN(Selection_basis);
      FOR counter:=1 TO 3 DO Selection_basis[counter]:=
                  UPCASE(Selection_basis[counter]);
    UNTIL (Selection_basis='A')
       OR (Selection_basis='B');
    IF Selection_basis='A' THEN
      BEGIN
      WRITELN;
      WRITELN
      ('Enter the number of plants to be selected each generation.');
      WRITELN
      ('Integer must not exceed 200, or the population size: ',population_size);
      WRITELN;
      number_selected:=
        Accept_integer('Number to be selected = ',1,Population_size);
      selection_pressure:=(1-(number_selected/population_size));
      WRITELN;
      WRITELN
      ('Selecting ',number_selected,' plants per generation equates to a');
      WRITELN
      ('selection pressure of ',selection_pressure:4:2);
      END
    ELSE IF Selection_basis='B' THEN
      BEGIN
      WRITELN;
      Max_selection_pressure:=1-(1/population_size);
      WRITELN
      ('Enter the selection pressure (as defined above), which can be any.');
      WRITELN
      ('real number between 0 and ',Max_selection_pressure:4:2);
      WRITELN;
      Selection_pressure:=
        Accept_real('Selection pressure = ',0,Max_selection_pressure);
      number_selected:=ROUND(population_size*(1-selection_pressure));
      corrected_selection_pressure:=1-(number_selected/population_size);
      selection_pressure:=corrected_selection_pressure;
      WRITELN;
      WRITELN
      ('The specified selection pressure rounds to the selection of ');
      WRITELN
      (number_selected,' plants per generation. Correcting for rounding');
      WRITELN
      ('errors, this gives a realised selection pressure of
',corrected_selection_pressure:4:2);
      END;

    WRITELN;
    WRITELN
    ('If these values are acceptable, Enter Y to proceed with the entry');
    WRITELN
    ('of the selection parameters.');
    WRITELN
    ('If you wish to repeat this section, enter N.');
    REPEAT
      WRITE
      ('Enter Y to proceed with the simulation, or N to repeat this section.');
      READLN(Selection_parameters_approved);
      FOR counter:=1 TO 3 DO Selection_parameters_approved[counter]:=
                  UPCASE(Selection_parameters_approved[counter]);
    UNTIL (Selection_parameters_approved='Y')
       OR (Selection_parameters_approved='N');
```

```
    END;
  UNTIL
    Selection_parameters_approved='Y';
  END;


PROCEDURE Ask_Selection_limit;
  BEGIN
  CLRSCR;
  WRITELN
  ('Enter the number of generations of selection. The maximum permitted');
  WRITELN
  ('value is 20. As a rough guide, under selfing, residual population');
  WRITELN
  ('heterozygosity is expected to be less than 1% after 7 generations.');
  WRITELN;
  Selection_limit:=
      Accept_integer('Number of generations of selection = ',1,20);
  WRITELN;
  CLRSCR;
  END;

{
PROCEDURE Ask_Mutation;
  VAR
    Answer: CHAR;
  BEGIN
  REPEAT
    WRITELN;
    WRITE
    ('Do you wish to consider mutation within the simulation? Y/N ');
    READLN(Answer);
  UNTIL (Answer='Y')
     OR (Answer='y')
     OR (Answer='N')
     OR (Answer='n');
  IF (Answer='Y') OR (Answer='y') THEN Mutation:=TRUE
  ELSE Mutation:=FALSE;
  CLRSCR;
  END;


PROCEDURE Ask_Migration;
  VAR
    Answer: CHAR;
  BEGIN
  REPEAT
    WRITELN;
    WRITE
    ('Do you wish to simulate migration within the simulation? Y/N ');
    READLN(Answer);
  UNTIL (Answer='Y')
     OR (Answer='y')
     OR (Answer='N')
     OR (Answer='n');
  IF (Answer='Y') OR (Answer='y') THEN
    BEGIN
    Migration:=TRUE;
    WRITELN;
    Migrants:=Accept_integer
              ('Mean number of migrants per generation = ',1,Population_size-1);
    Migration_frequency:=Migrants/Population_size;
    END;
  ELSE Migration:=FALSE;
  CLRSCR;
  END;
}
```

```
PROCEDURE Ask_Num_of_repetitions;
  BEGIN
  WRITELN
  ('Number of experimental repetitions (iterations of the program).');
  WRITELN
  ('The value must not exceed 50.');
  WRITELN;
  num_of_repetitions:=Accept_integer('Number of repetitions = ',1,50);
  WRITELN;
  CLRSCR;
  END;


PROCEDURE Program_specified_variables;
  BEGIN
  genotype_background:=ROUND(((A1A1+A2A2)/2)*num_of_loci);
                      { The product of intermediate A1A1 to A2A2 value
                        and number of loci }
  breeders:=number_selected;
  offspring_contribution:=population_size DIV number_selected;
  degrees_of_freedom:=num_of_repetitions-1;
  END;
{ Comment: Procedure "Variables" initialises all global variable values
           that are not specified diectly by the user }


PROCEDURE Front_screen;
  BEGIN
  TEXTBACKGROUND(BLUE);
  TEXTCOLOR(WHITE);
  CLRSCR;
  WRITELN
  ('Program PSELECT.PAS, December 1992.');
  WRITELN
  ('COMPUTER SIMULATION OF SELECTION IN A HYPOTHETICAL CROP SPECIES');
  WRITELN
  ('Paul Partner');
  WRITELN
  ('Institute of Ecology & Resource Management, University of Edinburgh');
  WRITELN;
  WRITELN;
  WRITELN
  ('The model simulates the effect of phenoypic truncation selection on');
  WRITELN
  ('an idealised quantitative character. Selection parameters, such as');
  WRITELN
  ('the number of loci defining the character, the allelic interaction at');
  WRITELN
  ('the loci, the breeding regime, population size and the intensity of');
  WRITELN
  ('selection, are specified by the user. The population response to');
  WRITELN
  ('selection is then simulated for the given parameters.');
  WRITELN;
  WRITELN
  ('This model deals exclusively with the action of selection upon the');
  WRITELN
  ('population. For purposes of simplification, all other systematic');
  WRITELN
  ('evolutionary forces are discounted.');
  WRITELN;
  WRITELN
  ('PLEASE NOTE: this model is mechanistic, and is not intended for use');
  WRITELN
  ('in a predictive or decision-support capacity.');
  WRITELN;
  REPEAT
    WRITE
    ('Enter P to proceed with the simulation, or Q to quit ');
```

```
    READLN(Quit_program);
    FOR counter:=1 TO 3 DO Quit_program[counter]:=
                    UPCASE(Quit_program[counter]);
  UNTIL (Quit_program='P')
     OR (Quit_program='Q');
  END;


PROCEDURE User_specified_variables;
  VAR
    Summary_approved:STRING[3]; { local variable to OK or reject values }
  BEGIN
  TEXTBACKGROUND(BLUE);
  TEXTCOLOR(WHITE);
  REPEAT
    BEGIN
    CLRSCR;
    Ask_Population_size;
    Ask_Number_of_loci;
    Ask_Allelic_action;
    Ask_Breeding_regime;
    Ask_Environmental_sd;
    Ask_Transformation_method;
    Ask_Selection_basis;
    Ask_Selection_limit;
{
    Ask_mutation;
    Ask_migration;
}
    Ask_Num_of_repetitions;

    WRITELN('Program PSELECT.PAS, December 1992.');
    WRITELN
    ('COMPUTER SIMULATION OF SELECTION IN A HYPOTHETICAL CROP SPECIES');
    WRITELN;
    WRITELN('    SUMMARY OF USER-DEFINED PARAMETERS');
    WRITELN;
    WRITELN('    Population size = ',population_size);
    WRITELN('    Number of loci defining character = ',num_of_loci);

    IF gene_action='DOM'
      THEN WRITELN
      ('    Full allelic dominance at all loci')
    ELSE
    IF gene_action='ADD'
      THEN WRITELN
      ('    Additive allelic action at all loci')
    ELSE
    IF gene_action='PAR'
      THEN WRITELN
      ('    Partial allelic dominance at all loci. Heterozygote =
',heterozygote_value:4:2);
    IF pollination_mode='MONO'
      THEN WRITELN('    Monogamous outbreeding')
    ELSE
    IF pollination_mode='POLY'
      THEN WRITELN('    Polygamous outbreeding')
    ELSE
    IF pollination_mode='SELF'
      THEN WRITELN('    Full inbreeding');

    WRITELN
    ('    Applied environmental standard deviation = ',environmental_sd:5:2);

    IF normal_method='BM'
      THEN WRITELN
      ('    Environment generated by Box-Muller transformation')
    ELSE
    IF normal_method='POLAR'
```

```
      THEN WRITELN
      ('    Environment generated by Marsaglia-Bray Polar transformation')
    ELSE
    IF normal_method='CLT'
      THEN WRITELN
      ('    Environment generated by Central Limit Theorem transformation');
    WRITELN
    ('    Number of plants to be selected each generation = ',number_selected);
    WRITELN
    ('    This equates to a selection pressure of ',selection_pressure:4:2);
    WRITELN
    ('    Selection to be practiced for ',selection_limit,' generations');
    WRITELN
    ('    Number of experimental repetitions = ',num_of_repetitions);
    WRITELN;
    WRITELN;

    WRITELN
    ('If the above details are correct, enter Y to proceed with simulation');
    WRITELN
    ('Enter any other character to return to parameter input procedure');
    WRITELN;

    REPEAT
      WRITE('    PROCEED WITH SIMULATION? ');
      READLN(Summary_approved);
      FOR counter:=1 TO 3 DO Summary_approved[counter]:=
                  UPCASE(Summary_approved[counter]);
    UNTIL (Summary_approved='Y')
       OR (Summary_approved='YES')
       OR (Summary_approved='N')
       OR (Summary_approved='NO');

    END;

  UNTIL (Summary_approved='Y')
     OR (Summary_approved='YES');
  END;


PROCEDURE F1_Population;
  BEGIN
  FOR plant:=1 to population_size DO
    BEGIN
    FOR locus:=1 to num_of_loci DO
      BEGIN
      population[plant,locus]:=A1A2;
      END;
    END;
  END;
{ Comment: Procedure "F1_Population" generates a population array
           where every locus is heterozygous A1A2, and hence has the
           value A1A2 }

{
PROCEDURE Mutations;
  BEGIN
  FOR Parent:=1 TO Breeders DO
    FOR Locus:=1 TO Num_of_loci DO
      IF RANDOM(20000)=1 THEN Parents[plant,locus]:=A2A2;
  END;


PROCEDURE Migrations;
  BEGIN
  RReal1:=(RANDOM(1000)+1)/1000;
  FOR Plant:=1 TO Population_size DO
    IF (RReal1 < Migration_frequency THEN
      BEGIN
```

```
      IF (RReal1 MOD 2 = 1) THEN
      FOR Locus:=1 TO Num_of_loci DO Population[Plant,Locus]:=A1A1;
      END
    ELSE
      FOR Locus:=1 TO Num_of_loci DO Population[Plant,Locus]:=A2A2;
  END;
}

PROCEDURE Population_Statistics;
  VAR
    cycle: INTEGER;
    i:INTEGER;
    n:INTEGER;
    RDev1,RDev2,sum_RReal1:REAL;
    T,u1,u2,u3,v1,v2:REAL;
    sdev_modifier:REAL;                    { to correct Box-Muller s.d.
                                             to sdev }
    environmental_mean_correction:REAL;    { to correct Box-Muller mean
                                             to zero }

  BEGIN
    sum_genotype:=0;
    ssq_genotype:=0;
    sum_phenotype:=0;
    ssq_phenotype:=0;
    sum_environment:=0;
    ssq_environment:=0;
    sum_RReal1:=0;
    FOR plant:=1 to population_size DO
      BEGIN
      genotype:=genotype_background;
      FOR locus:=1 to num_of_loci DO
        BEGIN
        CORRECT_FOR_DOMINANCE;
        genotype:=genotype+population[plant,locus];
        END;

      IF normal_method='CLT' THEN    { Central Limits Theorem }
        BEGIN
        sum_RReal1:=0;
        n:=12;
        FOR i:=1 TO n DO
          BEGIN
          RReal1:=(RANDOM(1000)+1)/1000;
          sum_RReal1:=RReal1+sum_RReal1;  {where final sum_RReal gives T}
          END;
        T:=sum_RReal1;
        environment:=(environmental_sd/(SQRT(n/12)))*(T-(n/2));
        END;

      IF normal_method='POLAR' THEN    { Marsaglia-Bray Polar transformation }
        BEGIN
          CASE (RANDOM(100)+1) OF
          1..86   : BEGIN
                    u1:=(RANDOM(1000)+1)/1000;
                    u2:=(RANDOM(1000)+1)/1000;
                    u3:=(RANDOM(1000)+1)/1000;
                    environment:=environmental_sd*2*(u1+u2+u3-1.5);
                    END;
          87..97  : BEGIN
                    u1:=(RANDOM(1000)+1)/1000;
                    u2:=(RANDOM(1000)+1)/1000;
                    environment:=environmental_sd*1.5*(u1+u2-1);
                    END;
          98..100 : BEGIN
                    REPEAT
                      v1:=(RANDOM(1001)/500)-1;
                      v2:=(RANDOM(1001)/500)-1;
                    UNTIL ( SQR(v1)+SQR(v2) < 1 );
```

```
                    environment:=environmental_sd*v1
                                 *SQRT(-2*ln(SQR(v1)+SQR(v2))
                                 /(SQR(v1)+SQR(v2)));
                END;
        ELSE WRITELN
        ('Something strange has happened in the CASE statement!');
        END;
      END;

    IF normal_method='BM' THEN    { Box-Muller transformation }
      BEGIN
      RDev1:=(RANDOM(1000)+1)/1000;
      RDev2:=(RANDOM(1000)+1)/1000;
      IF (Counter MOD 2 = 1) THEN
        environment:=(environmental_sd*(SQRT(-2*ln(RDev1))
                     *(COS(6.2831*(RDev2)))))
      ELSE
        environment:=(environmental_sd*(SQRT(-2*ln(RDev1))
                     *(SIN(6.2831*(RDev2)))))
      END;

    phenotype:=genotype+environment;
    population_phenotypes[plant]:=phenotype;
    sum_genotype:=sum_genotype+genotype;
    ssq_genotype:=ssq_genotype+SQR(genotype);
    sum_phenotype:=sum_phenotype+phenotype;
    ssq_phenotype:=ssq_phenotype+SQR(phenotype);
    sum_environment:=sum_environment+environment;
    ssq_environment:=ssq_environment+SQR(environment);
    END;
  mean_genotype:=sum_genotype/population_size;
  mean_phenotype:=sum_phenotype/population_size;
  IF Population_size=1 THEN
    BEGIN
    var_genotype:=0;
    var_phenotype:=0;
    var_environment:=0;
    END
  ELSE
    BEGIN
    var_genotype:=(ssq_genotype-sqr(sum_genotype)/population_size)
                  /(population_size-1);
    var_phenotype:=(ssq_phenotype-sqr(sum_phenotype)/population_size)
                   /(population_size-1);
    var_environment:=(ssq_environment-SQR(sum_environment)/population_size)
                     /(population_size-1);
    H2:=var_genotype/var_phenotype;
    END;
  mean_environment:=sum_environment/population_size;

  heterozygous:=0;
  FOR plant:=1 to population_size DO
    BEGIN
    FOR locus:=1 to num_of_loci DO
      BEGIN
      IF population[plant,locus]=A1A2 THEN heterozygous:=heterozygous+1;
      END;
    heterozygosity:=heterozygous/(num_of_loci*population_size);
    END;

  CALCULATE_ALLELIC_FREQUENCIES;

  END;
{ Comment: Procedure "Population_Statistics" calculates the population
           statistics for the generation }
```

```
PROCEDURE Store_population_stats_for_this_repetition;
  BEGIN
  IF Debug_on
  THEN
    BEGIN
    APPEND(DBug_output);
    WRITELN(Dbug_output,'Repetition ',repetition:4);
    WRITELN(Dbug_output,'Generation ',generation:4);
    WRITELN(Dbug_output,'P mean_genotype ',mean_genotype:7:1);
    WRITELN(Dbug_output,'P var_genotype ',var_genotype:6:1);
    WRITELN(DBug_output,'P mean_environment ',mean_environment:7:1);
    WRITELN(DBug_output,'P var_environment ',var_environment:6:1);
    WRITELN(Dbug_output,'P mean_phenotype ',mean_phenotype:7:1);
    WRITELN(Dbug_output,'P var_phenotype ',var_phenotype:6:1);
    WRITELN(Dbug_output,'P heterozygosity ',heterozygosity:6:3);
    END;
  results[repetition,generation,1]:=mean_genotype;
  results[repetition,generation,2]:=var_genotype;
  results[repetition,generation,3]:=mean_environment;
  results[repetition,generation,4]:=var_environment;
  results[repetition,generation,5]:=mean_phenotype;
  results[repetition,generation,6]:=var_phenotype;
  results[repetition,generation,7]:=heterozygosity;
  results[repetition,generation,8]:=p;
  results[repetition,generation,9]:=q;
  END;
{ Comment: This procedure stores the data output for each repetition
           in a 3-dimensional array, called results }


PROCEDURE Selection_of_breeders;
  BEGIN
{
  IF Mutation THEN Mutations;
}
  FOR parent:=1 to breeders DO
    BEGIN
    TempReal:=population_phenotypes[1];
    TempInt:=1;
    FOR plant:=2 to population_size DO
      BEGIN
      IF Selection_direction='UP' THEN
        BEGIN
        IF population_phenotypes[plant]>TempReal THEN
          BEGIN
          TempReal:=population_phenotypes[plant];
          TempInt:=plant;
          END;
        END;
      IF Selection_Direction='DOWN' THEN
        BEGIN
        IF population_phenotypes[plant]<TempReal THEN
          BEGIN
          TempReal:=population_phenotypes[plant];
          TempInt:=plant;
          END;
        END;
      FOR locus:=1 to num_of_loci DO
        BEGIN
        parents[parent,locus]:=population[TempInt,locus];
        END;
      parental_phenotypes[parent]:=population_phenotypes[TempInt];
      population_phenotypes[TempInt]:=0;
    END;

  IF Debug_on
  THEN
    BEGIN
```

```
    APPEND(DBug_output);
    FOR parent:=1 to breeders DO
      BEGIN
      FOR locus:=1 to num_of_loci DO
        BEGIN
        WRITE(DBug_output,parents[parent,locus]:3);
        END;
      WRITELN(DBug_output,' ');
      END;
    WRITELN(DBug_output,'Procedure "Selection" has been executed.');
    WRITELN(DBug_output,' ');
    CLOSE(DBug_output);
    END;

  END;
{ Comment: Procedure "Selection_of_breeders" identifies and isolates the
           best plants, and generates a selected parents array }


PROCEDURE Selected_Statistics;
  BEGIN
  sum_genotype:=0;
  ssq_genotype:=0;
  sum_phenotype:=0;
  ssq_phenotype:=0;
  FOR parent:=1 to breeders DO
    BEGIN
    genotype:=genotype_background;
    FOR locus:=1 to num_of_loci DO
      BEGIN
      CORRECT_FOR_DOMINANCE;
      genotype:=genotype+parents[parent,locus];
      END;
    sum_genotype:=sum_genotype+genotype;
    ssq_genotype:=ssq_genotype+sqr(genotype);
    sum_phenotype:=sum_phenotype+parental_phenotypes[parent];
    ssq_phenotype:=ssq_phenotype+sqr(parental_phenotypes[parent]);
    END;
  mean_genotype:=sum_genotype/breeders;
  mean_phenotype:=sum_phenotype/breeders;
  IF breeders=1 THEN
    BEGIN
    var_genotype:=0;
    var_phenotype:=0;
    END
  ELSE
    BEGIN
    var_genotype:=(ssq_genotype-sqr(sum_genotype)/breeders)/(breeders-1);
    var_phenotype:=(ssq_phenotype-sqr(sum_phenotype)/breeders)/(breeders-1);
    END;
  heterozygous:=0;
  FOR parent:=1 to breeders DO
    BEGIN
    FOR locus:=1 to num_of_loci DO
      BEGIN
      IF parents[parent,locus]=A1A2 THEN heterozygous:=heterozygous+1;
      END;
    heterozygosity:=heterozygous/(breeders*num_of_loci);
    END;
  END;
{ Comment: Procedure "Selected Statistics" calculates the statistics
           for the selected plants array }


PROCEDURE Store_selected_stats_for_this_repetition;
  BEGIN

  IF Debug_on
  THEN
```

```
    BEGIN
    APPEND(DBug_output);
    WRITELN(Dbug_output,'S mean_genotype ',mean_genotype:7:1);
    WRITELN(Dbug_output,'S var_genotype ',var_genotype:6:1);
    WRITELN(Dbug_output,'S mean_phenotype ',mean_phenotype:7:1);
    WRITELN(Dbug_output,'S var_phenotype ',var_phenotype:6:1);
    WRITELN(Dbug_output,'S heterozygosity ',heterozygosity:6:3);
    WRITELN(Dbug_output,' ');
    END;

  results[repetition,generation,10]:=mean_genotype;
  results[repetition,generation,11]:=var_genotype;
  results[repetition,generation,12]:=mean_phenotype;
  results[repetition,generation,13]:=var_phenotype;
  results[repetition,generation,14]:=heterozygosity;
  END;
{ Comment: This procedure stores the data output for each repetition
           in a 3-dimensional array, called results }


PROCEDURE Population_Array;
  BEGIN

{ Comment: "offspring_contribution" will always be the inverse of
           number selected.
           offspring_contribution:=population_size/number_selected }

{ Comment: "offspring" is the specific reference pointer to the individual
           plant among the overall contribution of the parent to next
           generation. }

{ Comment: "plant:=(offspring_contribution*(parent-1))+offspring" allows
           reconstitution of the population array in blocks of "contribution"
           from the parent array. }

{ Comment: In order to allow the expansion of this model to cope with
           selection in OUTBREEDERS, this procedure should be coded into
           separate routines for (1) POLLINATION within the selected parents,
           (2) MULTIPLICATION (bulking) of each parent to reconstitute the
           [1..population_size] POPULATION array.

{ Comment: Routine for bulking each parent to n=offspring_contribution
           offspring }
{
    IF Migration THEN Migrations;
}
  IF pollination_mode='MONO' THEN
    BEGIN
    FOR parent:=1 TO breeders DO  {sequential specification of female parent}
      BEGIN
      female:=parent;
      REPEAT male:=ROUND(RANDOM(breeders+1)) UNTIL male<>female;
      FOR offspring:=1 TO offspring_contribution DO
        BEGIN
        FOR locus:=1 TO num_of_loci DO
          BEGIN
          plant:=(offspring_contribution*(parent-1))+offspring;

            CASE parents[female,locus] OF
            A1A1: CASE parents[male,locus] OF
                  A1A1: population[plant,locus]:=A1A1;
                  A1A2: BEGIN
                        allele_segregation:=(RANDOM(100)+1);
                        IF allele_segregation<=50
                        THEN population[plant,locus]:=A1A1
                        ELSE population[plant,locus]:=A1A2;
                        END;
                  A2A2: population[plant,locus]:=A1A2
                  END;
```

```
          A1A2: CASE parents[male,locus] OF
                A1A1: BEGIN
                      allele_segregation:=(RANDOM(100)+1);
                      IF allele_segregation<=50
                      THEN population[plant,locus]:=A1A1
                      ELSE population[plant,locus]:=A1A2;
                      END;
                A1A2: BEGIN
                      allele_segregation:=(RANDOM(100)+1);
                      IF allele_segregation<=25
                      THEN population[plant,locus]:=A1A1
                      ELSE
                      IF allele_segregation>75
                      THEN population[plant,locus]:=A2A2
                      ELSE population[plant,locus]:=A1A2;
                      END;
                A2A2: BEGIN
                      allele_segregation:=(RANDOM(100)+1);
                      IF allele_segregation<=50
                      THEN population[plant,locus]:=A1A2
                      ELSE population[plant,locus]:=A2A2;
                      END;
                END;
          A2A2: CASE parents[male,locus] OF
                A1A1: population[plant,locus]:=A1A2;
                A1A2: BEGIN
                      allele_segregation:=(RANDOM(100)+1);
                      IF allele_segregation<=50
                      THEN population[plant,locus]:=A1A2
                      ELSE population[plant,locus]:=A2A2;
                      END;
                A2A2: population[plant,locus]:=A2A2
                END;
          END;

        END;
      END;
    END;
  END;

IF pollination_mode='POLY' THEN
  BEGIN
  FOR parent:=1 TO breeders DO  {sequential specification of female parent}
    BEGIN
    female:=parent;
    FOR offspring:=1 TO offspring_contribution DO
      BEGIN
      REPEAT male:=ROUND(RANDOM(breeders+1)) UNTIL male<>female;
      FOR locus:=1 TO num_of_loci DO
        BEGIN
        plant:=(offspring_contribution*(parent-1))+offspring;

          CASE parents[female,locus] OF
          A1A1: CASE parents[male,locus] OF
                A1A1: population[plant,locus]:=A1A1;
                A1A2: BEGIN
                      allele_segregation:=(RANDOM(100)+1);
                      IF allele_segregation<=50
                      THEN population[plant,locus]:=A1A1
                      ELSE population[plant,locus]:=A1A2;
                      END;
                A2A2: population[plant,locus]:=A1A2
                END;
          A1A2: CASE parents[male,locus] OF
                A1A1: BEGIN
                      allele_segregation:=(RANDOM(100)+1);
                      IF allele_segregation<=50
                      THEN population[plant,locus]:=A1A1
                      ELSE population[plant,locus]:=A1A2;
```

```
                    END;
              A1A2: BEGIN
                    allele_segregation:=(RANDOM(100)+1);
                    IF allele_segregation<=25
                    THEN population[plant,locus]:=A1A1
                    ELSE IF allele_segregation>75
                    THEN population[plant,locus]:=A2A2
                    ELSE population[plant,locus]:=A1A2;
                    END;
              A2A2: BEGIN
                    allele_segregation:=(RANDOM(100)+1);
                    IF allele_segregation<=50
                    THEN population[plant,locus]:=A1A2
                    ELSE population[plant,locus]:=A2A2;
                    END;
              END;
        A2A2: CASE parents[male,locus] OF
              A1A1: population[plant,locus]:=A1A2;
              A1A2: BEGIN
                    allele_segregation:=(RANDOM(100)+1);
                    IF allele_segregation<=50
                    THEN population[plant,locus]:=A1A2
                    ELSE population[plant,locus]:=A2A2;
                    END;
              A2A2: population[plant,locus]:=A2A2
              END;
        END;

      END;
    END;
  END;

IF pollination_mode='SELF' THEN
  BEGIN
  FOR parent:=1 TO breeders DO
    BEGIN
    FOR offspring:=1 TO offspring_contribution DO
      BEGIN
      FOR locus:=1 TO num_of_loci DO
        BEGIN
        plant:=(offspring_contribution*(parent-1))+offspring;
        population[plant,locus]:=parents[parent,locus];

{ Comment: Routine for specification of allelic segregation in
           population array }

        CASE population[plant,locus] OF
        A1A1: population[plant,locus]:=A1A1;
        A1A2: BEGIN
              allele_segregation:=(RANDOM(100)+1);
              IF allele_segregation<=25
              THEN population[plant,locus]:=A1A1
              ELSE
              IF allele_segregation>75
              THEN population[plant,locus]:=A2A2
              ELSE population[plant,locus]:=A1A2;
              END;
        A2A2: population[plant,locus]:=A2A2;
        END;

        END;
      END;
    END;

  IF Debug_on
  THEN
    BEGIN
    APPEND(DBug_output);
```

```
      WRITELN(DBug_output,'Procedure "Population_Array" entered');
      WRITELN(DBug_output,' ');
      FOR plant:=1 to population_size DO
        BEGIN
        FOR locus:=1 to num_of_loci DO
          BEGIN
          WRITE(DBug_output,population[plant,locus]:3);
          END;
        WRITELN(DBug_output,' ');
        END;
      WRITELN(DBug_output,'Procedure "Population Array" has been executed.');
      WRITELN(DBug_output,' ');
      CLOSE(DBug_output);
      END;
    END;
  END;
{ Comment: Procedure "Population Array" generates the population array for
           generation F2 and subsequent generations. CASE section 1 is for
           inbreeding, while CASE section 2 deals with random outbreeding
           among selected plants }


{ Comment: Routine to calculate allelic frequencies from the
  POPULATION and/or PARENTS array.
  A decision will need to be made as to the timing of mutation
  events: mutation at the gametic stage vs. mutation within the
  mature population }


PROCEDURE Output_results_of_this_repetition_to_file;
  BEGIN
  IF (repetition=1) THEN
    BEGIN
    ASSIGN(data_output,'PSELECT.RES');
    REWRITE(data_output);
    WRITELN(data_output,'Program PSELECT.PAS');
    WRITELN(data_output);
    WRITELN
    (data_output,'Response to selection in a hypothetical plant species');
    WRITELN(data_output);
    WRITELN(data_output,'  Population                                    Selected
plants');
    WRITELN(data_output);
    WRITELN(data_output,'F Genotype  Environment Phenotype Het  P  Q  Genotype
Phenotype    Het');
    WRITELN(data_output,'  MEAN V(G) MEAN V(E)   MEAN V(P)            MEAN  V(G)
MEAN  V(P)');
    END
  ELSE APPEND(data_output);
  WRITELN(data_output);
  WRITELN(data_output,'Repetition No. ',repetition);
  WRITELN(data_output);
  FOR gen:=1 TO selection_limit DO
    BEGIN
    WRITE(data_output,gen:1);
    WRITE(data_output,results[repetition,gen,1]:6:1);
    WRITE(data_output,results[repetition,gen,2]:6:1);
    WRITE(data_output,results[repetition,gen,3]:7:1);
    WRITE(data_output,results[repetition,gen,4]:6:1);
    WRITE(data_output,results[repetition,gen,5]:7:1);
    WRITE(data_output,results[repetition,gen,6]:6:1);
    WRITE(data_output,results[repetition,gen,7]:6:3);
    WRITE(data_output,results[repetition,gen,8]:6:2);
    WRITE(data_output,results[repetition,gen,9]:6:2);
    WRITE(data_output,results[repetition,gen,10]:7:1);
    WRITE(data_output,results[repetition,gen,11]:6:1);
    WRITE(data_output,results[repetition,gen,12]:6:3);
{   WRITE(data_output,results[repetition,gen,13]:6:2);   }
    WRITELN(data_output);
```

```
    END;
  CLOSE(data_output);
  END;
{ Comment: this procedure writes simulation data to a listing of results
           from each simulation run. (File: filename.AGG) }


PROCEDURE Calculate_aggregate_results;
  BEGIN
  sqrt_of_num_of_reps:=SQRT(num_of_repetitions);
  FOR gen:=1 TO selection_limit DO
    BEGIN
    FOR population_value:=1 TO num_of_population_values DO
      BEGIN
      sum_of_repetitions:=0;
      ssq_of_repetitions:=0;
      FOR repetition:=1 TO num_of_repetitions DO
        BEGIN
        sum_of_repetitions:=
            sum_of_repetitions+results[repetition,gen,population_value];
        ssq_of_repetitions:=
            ssq_of_repetitions+SQR(results[repetition,gen,population_value]);
        END;
      mean_of_repetitions[gen,population_value]:=
          sum_of_repetitions/num_of_repetitions;
      sd_of_repetitions[gen,population_value]:=
          SQRT((ssq_of_repetitions-SQR(sum_of_repetitions)/num_of_repetitions)
          /(num_of_repetitions-1));
      sem_of_repetitions[gen,population_value]:=
          sd_of_repetitions[gen,population_value]/sqrt_of_num_of_reps;
      END;
    END;
  END;
{ Comment: this procedure calulates the mean across the repetitions of the
           simulated runs. }


PROCEDURE Output_aggregated_results_to_file;
  BEGIN
  ASSIGN(aggr_output,'PSELECT.AGG');
  REWRITE(aggr_output);
  WRITELN(aggr_output,'Program PSELECT.PAS');
  WRITELN(aggr_output);
  WRITELN(aggr_output,'Response to selection in a hypothetical plant species');
  WRITELN(aggr_output);
  WRITELN(aggr_output,'   Population                                        Selected
plants');
  WRITELN(aggr_output);
  WRITELN(aggr_output,'F  Genotype     Environment  Phenotype    Het   Genotype
Phenotype    Het');
  WRITELN(aggr_output,'   MEAN  V(G)   MEAN  V(E)   MEAN  V(P)         MEAN  V(G)
MEAN  V(P)');
  WRITELN(aggr_output);

  IF num_of_repetitions=1 THEN
    BEGIN
    FOR gen:=1 TO selection_limit DO
      BEGIN
      WRITE(aggr_output,gen:1);
      WRITE(aggr_output,results[repetition,gen,1]:6:1);
      WRITE(aggr_output,results[repetition,gen,2]:6:1);
      WRITE(aggr_output,results[repetition,gen,3]:7:1);
      WRITE(aggr_output,results[repetition,gen,4]:6:1);
      WRITE(aggr_output,results[repetition,gen,5]:7:1);
      WRITE(aggr_output,results[repetition,gen,6]:6:1);
      WRITE(aggr_output,results[repetition,gen,7]:6:2);
      WRITE(aggr_output,results[repetition,gen,8]:7:1);
      WRITE(aggr_output,results[repetition,gen,9]:6:1);
      WRITE(aggr_output,results[repetition,gen,10]:7:1);
```

```
      WRITE(aggr_output,results[repetition,gen,11]:6:1);
      WRITE(aggr_output,results[repetition,gen,12]:6:2);
{     WRITE(aggr_output,results[repetition,gen,13]:6:2); }
      WRITELN(aggr_output);
      END;
    END
  ELSE IF num_of_repetitions>1 THEN
  FOR gen:=1 TO selection_limit DO
    BEGIN
    WRITE(aggr_output,gen:1);
    WRITE(aggr_output,mean_of_repetitions[gen,1]:6:1);
    WRITE(aggr_output,mean_of_repetitions[gen,2]:6:1);
    WRITE(aggr_output,mean_of_repetitions[gen,3]:7:1);
    WRITE(aggr_output,mean_of_repetitions[gen,4]:6:1);
    WRITE(aggr_output,mean_of_repetitions[gen,5]:7:1);
    WRITE(aggr_output,mean_of_repetitions[gen,6]:6:1);
    WRITE(aggr_output,mean_of_repetitions[gen,7]:6:2);
    WRITE(aggr_output,mean_of_repetitions[gen,8]:7:1);
    WRITE(aggr_output,mean_of_repetitions[gen,9]:6:1);
    WRITE(aggr_output,mean_of_repetitions[gen,10]:7:1);
    WRITE(aggr_output,mean_of_repetitions[gen,11]:6:1);
    WRITE(aggr_output,mean_of_repetitions[gen,12]:6:2);
{   WRITE(aggr_output,mean_of_repetitions[gen,13]:6:2);   }
    WRITELN(aggr_output);
    END;

  WRITELN(aggr_output);
  WRITE(aggr_output,'Population size = ');
  WRITE(aggr_output,population_size:3);
  WRITELN(aggr_output);

  IF Pollination_mode='MONO'
    THEN WRITE(aggr_output,'Monogamous random outcrossing')
    ELSE
  IF Pollination_mode='POLY'
    THEN WRITE(aggr_output,'Polygamous random outcrossing')
    ELSE
  IF Pollination_mode='SELF'
    THEN WRITE(aggr_output,'Full inbreeding')
    ELSE
  WRITE(aggr_output,'It is breeding strangely!');
  WRITELN(aggr_output);

  IF gene_action='DOM'
    THEN WRITE(aggr_output,'Full allelic dominance at ')
    ELSE
  IF gene_action='ADD'
    THEN WRITE(aggr_output,'Additive allelic action at ')
    ELSE
  IF gene_action='PAR'
    THEN WRITE(aggr_output,'Allelic action as specified at ');
  WRITE(aggr_output,num_of_loci:2);
  WRITE(aggr_output,' loci');
  WRITELN(aggr_output);

  WRITE(aggr_output,'Genotypic background = ');
  WRITE(aggr_output,genotype_background:2);
  WRITELN(aggr_output);
  WRITE(aggr_output,'Selection pressure = ');
  WRITE(aggr_output,selection_pressure:4:2);
  WRITE(aggr_output,' per generation');
  WRITELN(aggr_output);
  WRITE(aggr_output,'Applied environmental standard deviation = ');
  WRITE(aggr_output,environmental_sd:5:2);
  WRITELN(aggr_output);

  WRITE(aggr_output,'Environmental generation method = ');
  IF normal_method='UBM'
    THEN WRITE(aggr_output,'Unmodified Box-Muller ')
```

```
    ELSE
  IF normal_method='MBM'
    THEN WRITE(aggr_output,'Modified Box-Muller ')
    ELSE
  IF normal_method='CLT'
    THEN WRITE(aggr_output,'Central Limit Theorem ')
    ELSE
  IF normal_method='POLAR'
    THEN WRITE(aggr_output,'Marsaglia-Bray Polar method ')
    ELSE
      WRITE(aggr_output,'Unspecified ');
  WRITELN(aggr_output);

  WRITE(aggr_output,'Number of repetitions = ');
  WRITE(aggr_output,num_of_repetitions:2);
  WRITELN(aggr_output);

  IF num_of_repetitions>1 THEN
    BEGIN
    WRITELN(aggr_output);
    WRITELN(aggr_output,'   Population                                          Selected
plants');
    WRITELN(aggr_output,'F  Genotype     Environment  Phenotype    Het   Genotype
Phenotype    Het');
    WRITELN(aggr_output,'   MEAN  V(G)   MEAN  V(E)   MEAN  V(P)         MEAN  V(G)
MEAN  V(P)');
    WRITELN(aggr_output);
    WRITELN(aggr_output,'   Standard deviations');
    WRITELN(aggr_output);

    FOR gen:=1 TO selection_limit DO
      BEGIN
      WRITE(aggr_output,gen:1);
      WRITE(aggr_output,sd_of_repetitions[gen,1]:6:2);
      WRITE(aggr_output,sd_of_repetitions[gen,2]:6:2);
      WRITE(aggr_output,sd_of_repetitions[gen,3]:7:2);
      WRITE(aggr_output,sd_of_repetitions[gen,4]:6:2);
      WRITE(aggr_output,sd_of_repetitions[gen,5]:7:2);
      WRITE(aggr_output,sd_of_repetitions[gen,6]:6:2);
      WRITE(aggr_output,sd_of_repetitions[gen,7]:6:2);
      WRITE(aggr_output,sd_of_repetitions[gen,8]:7:2);
      WRITE(aggr_output,sd_of_repetitions[gen,9]:6:2);
      WRITE(aggr_output,sd_of_repetitions[gen,10]:7:2);
      WRITE(aggr_output,sd_of_repetitions[gen,11]:6:2);
      WRITE(aggr_output,sd_of_repetitions[gen,12]:6:2);
{     WRITE(aggr_output,sd_of_repetitions[gen,13]:6:2);   }
      WRITELN(aggr_output);
      END;

    WRITELN(aggr_output);
    WRITELN(aggr_output,'   Standard errors');
    WRITELN(aggr_output);
    FOR gen:=1 TO selection_limit DO
      BEGIN
      WRITE(aggr_output,gen:1);
      WRITE(aggr_output,sem_of_repetitions[gen,1]:6:2);
      WRITE(aggr_output,sem_of_repetitions[gen,2]:6:2);
      WRITE(aggr_output,sem_of_repetitions[gen,3]:7:2);
      WRITE(aggr_output,sem_of_repetitions[gen,4]:6:2);
      WRITE(aggr_output,sem_of_repetitions[gen,5]:7:2);
      WRITE(aggr_output,sem_of_repetitions[gen,6]:6:2);
      WRITE(aggr_output,sem_of_repetitions[gen,7]:6:2);
      WRITE(aggr_output,sem_of_repetitions[gen,8]:7:2);
      WRITE(aggr_output,sem_of_repetitions[gen,9]:6:2);
      WRITE(aggr_output,sem_of_repetitions[gen,10]:7:2);
      WRITE(aggr_output,sem_of_repetitions[gen,11]:6:2);
      WRITE(aggr_output,sem_of_repetitions[gen,12]:6:2);
{     WRITE(aggr_output,sem_of_repetitions[gen,13]:6:2);   }
      WRITELN(aggr_output);
```

```
      END;

    WRITELN(aggr_output);
    WRITELN(aggr_output,'   Degrees of freedom = ',degrees_of_freedom);
    END;
  CLOSE(aggr_output);
  END;
{ Comment: this procedure outputs the simulated mean values to an
           aggregated data file: filename.AGG }


PROCEDURE Display_wait_screen;
  BEGIN
  TEXTBACKGROUND(RED);
  TEXTCOLOR(WHITE);
  CLRSCR;
  WRITELN;
  WRITELN
  (' Results are being written to the output file(s)');
  WRITELN
  (' Full execution can take several minutes. Please wait...');
  END;


PROCEDURE Exit_screen;
  BEGIN
  TEXTBACKGROUND(BLUE);
  TEXTCOLOR(WHITE);
  CLRSCR;
  IF num_of_repetitions>1 THEN
    BEGIN
    WRITELN;
    WRITELN
    ('The program has executed, and simulated results have been written');
    WRITELN
    ('to the files PSELECT.AGG (aggregated results across repetitions)');
    WRITELN
    ('and PSELECT.RES (results by repetition). These files should be');
    WRITELN
    ('accessed through DOS.');
    WRITELN;
    WRITE('Press ENTER key to exit program.');
    READ(Exit_program);
    END
  ELSE
    BEGIN
    WRITELN;
    WRITELN
    ('The program has executed, and simulated results have been written');
    WRITELN
    ('to the file PSELECT.RES. This file should be accessed through DOS.');
    WRITELN;
    WRITE('Press ENTER key to exit program.');
    READ(Exit_program);
    END;
  CLRSCR;
  END;


{ Comment: main program block }

  BEGIN
  Debug_on=FALSE;
  IF Debug_on THEN OPEN_DBUG_OUTPUT;
  FRONT_SCREEN;
  IF NOT (Quit_program='Q') THEN
    BEGIN
    USER_SPECIFIED_VARIABLES;
    Selection_direction:='UP';
```

```
    DISPLAY_WAIT_SCREEN;
    PROGRAM_SPECIFIED_VARIABLES;
    FOR repetition:=1 to num_of_repetitions DO
      BEGIN
      RANDOMIZE;
{     RANDSEED:=1;    }
      Generation:=1;

      F1_POPULATION;
      POPULATION_STATISTICS;
      Store_population_stats_for_this_repetition;
      SELECTION_OF_BREEDERS;
      SELECTED_STATISTICS;
      Store_selected_stats_for_this_repetition;

      FOR Generation:=2 to selection_limit DO
        BEGIN
        POPULATION_ARRAY;
        POPULATION_STATISTICS;
        Store_population_stats_for_this_repetition;
        SELECTION_OF_BREEDERS;
        SELECTED_STATISTICS;
        Store_selected_stats_for_this_repetition;
        END;
      Output_results_of_this_repetition_to_file;
      END;

    IF num_of_repetitions>1 THEN
      BEGIN
      Calculate_aggregate_results;
      END;

    Output_aggregated_results_to_file;
    EXIT_SCREEN;
    END;

  IF Debug_on THEN CLOSE_DBUG_OUTPUT;
  CLRSCR;
  WRITELN
  (' Thank you for using PSELECT. Press ENTER key to quit program');
  READ(Exit_program);

END.
```

APPENDIX B3

RTest04.pas program code

```
PROGRAM RTest04;
USES Crt,Dos;
CONST
  sd=1;
  Transformation_Method='BOX';  { values = NONE, CLT, MBP, BOX }
VAR
  Deviate: REAL;
  Data_Output: TEXT;
  Counter: INTEGER;

PROCEDURE Box_Muller;
VAR
  RDev1, RDev2: REAL;
BEGIN
RDev1:=(RANDOM(1000)+1)/1000;
RDev2:=(RANDOM(1000)+1)/1000;
IF (Counter MOD 2 = 1) THEN
  Deviate:=(sd*(SQRT(-2*ln(RDev1))*(COS(6.2831*(RDev2)))))
ELSE
  Deviate:=(sd*(SQRT(-2*ln(RDev1))*(SIN(6.2831*(RDev2)))));
END;

PROCEDURE Marsaglia_Bray;
VAR
  u1, u2, u3, v1, v2: REAL;
BEGIN
CASE (RANDOM(100)+1) OF
  1..86   : BEGIN
            u1:=(RANDOM(1000)+1)/1000;
            u2:=(RANDOM(1000)+1)/1000;
            u3:=(RANDOM(1000)+1)/1000;
            Deviate:=sd*2*(u1+u2+u3-1.5);
            END;
  87..97  : BEGIN
            u1:=(RANDOM(1000)+1)/1000;
            u2:=(RANDOM(1000)+1)/1000;
            Deviate:=sd*1.5*(u1+u2-1);
            END;
  98..100 : BEGIN
            REPEAT
              v1:=(RANDOM(1001)/500)-1;
              v2:=(RANDOM(1001)/500)-1;
            UNTIL ( SQR(v1)+SQR(v2) < 1 );
              Deviate:=sd*v1*SQRT(-2*ln(SQR(v1)+SQR(v2))/(SQR(v1)+SQR(v2)));
            END;
  END;
END;

PROCEDURE Central_Limits;
VAR
  Sum_RDev1,RDev1,T:REAL;
  n,i:INTEGER;
BEGIN
Sum_RDev1:=0;
n:=12;
FOR i:=1 TO n DO
  BEGIN
  RDev1:=(RANDOM(1000)+1)/1000;
  Sum_RDev1:=RDev1+sum_RDev1;
  END;
T:=Sum_RDev1;
Deviate:=(sd/(SQRT(n/12)))*(T-(n/2));
```

```
END;

BEGIN
RANDOMIZE;
ASSIGN(Data_Output,'A:RTest04.TXT');
REWRITE(Data_Output);
WRITELN(Data_Output,Transformation_Method);

FOR Counter:=1 to 100 DO
  BEGIN
  IF Transformation_Method='NONE'
  THEN Deviate:=(RANDOM(100)+1)
  ELSE
  IF Transformation_Method='CLT'
  THEN Central_Limits
  ELSE
  IF Transformation_Method='MBP'
  THEN Marsaglia_Bray
  ELSE
  IF Transformation_method='BOX'
  THEN Box_Muller;
  WRITELN(Data_Output,Deviate:6:2);
  END;
CLOSE(Data_Output);
END.
```

APPENDIX C1

Peat-based compost mixture used in selection programme

One 60 litre bail of sphagnum moss peat,
675g (+/-1g) dolomitic limestone dust,
675g (+/-1g) lime,
450g (+/-1g) single superphosphate,
120g (+/-1g) 34.5% ammonium nitrate (ICI *Nitram*),
120g (+/-1g) silicated trace elements (contains B, Cu, Fe, Mn, Mb, Zn)
120g (+/-1g) potassium nitrate.

APPENDIX C2

Sowing dates for *Brassica* spp. breeding material

Generation	Species	Material	Sown
P	*B. carinata*	Addis Aceb	12/2/90
		Chembere Dzagumhana	12/2/90
		Tamu Texel Greens	12/2/90
	B. juncea	J/1078/5/4/89/2013	12/2/90
		Stoke	12/2/90
		Trowse	12/2/90
F1	*B. carinata*	CA	28/8/90
		CB	28/8/90
		CC	28/8/90
		CD	28/8/90
		CH	28/8/90
		CK	28/8/90
	B. juncea	JA	28/8/90
		JB	28/8/90
		JC	28/8/90
		JD	28/8/90
		JH	28/8/90
		JK	28/8/90
F2	*B. carinata*	CC	27/3/91
		CD	27/3/91
		CH	27/3/91
		CK	27/3/91
	B. juncea	JA	28/3/91
		JB	28/3/91
		JC	28/3/91
		JD	28/3/91
		JH	28/3/91
		JK	28/3/91
F3	*B. carinata*	CC	2/3/92
		CD	2/3/92
		CH	2/3/92
		CK	2/3/92
	B. juncea	JA	4/3/92
		JB	4/3/92
		JC	5/3/92
		JD	5/3/92
		JH	4/3/92
		JK	4/3/92
F4	*B. carinata*	CC	2/9/92
		CD	2/9/92
		CH	2/9/92
		CK	2/9/92
	B. juncea	JA	30/8/92
		JB	30/8/92
		JC	30/8/92
		JD	30/8/92
		JH	30/8/92
		JK	30/8/92

APPENDIX C3

Population data for *Brassica carinata* and *Brassica juncea* crosses

C3.1 Brassica carinata parental populations

B.carinata parental population ADDIS ACEB

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	-1	2	960	11
2	2	0	3	1000	12
3	1	0	2	1085	12
4	1	0	2	1050	12
5	3	-2	2	995	13
6	2	-1	2	1020	12
7	1	0	2	1010	11
8	2	-1	3	1075	12
9	1	0	2	1090	11
10	3	-3	3	1010	13
11	3	-2	2	1065	12
12	1	0	2	1030	11
13	1	0	2	1105	12
14	2	-1	2	1035	14
15	2	0	2	1025	12
16	2	-2	2	980	12
17	2	0	2	1090	11
18	1	0	2	1120	12
19	3	-2	2	995	12
20	3	-2	2	980	11
n	20	20	20	20	20
Mean	1.90	-0.85	2.15	1036.00	11.90
Variance	0.621	0.976	0.134	2162.105	0.621
s.d.	0.788	0.988	0.366	46.498	0.788
s.e.m.	0.176	0.221	0.082	10.397	0.176

B.carinata parental population CHEMBERE DZAGUMHANA

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	2	1100	11
2	3	-2	2	990	12
3	3	-2	2	1010	12
4	2	0	2	1040	12
5	1	0	2	1110	13
6	3	-1	2	1020	12
7	2	0	3	1100	11
8	3	-2	2	1060	12
9	1	0	2	1100	11
10	3	-1	2	1030	13
11	4	-3	2	910	12
12	2	0	2	1100	11
13	3	-1	2	880	12
14	2	0	2	1020	14
15	2	0	3	1370	12
16	1	0	2	980	12
17	1	0	2	1040	11
18	2	0	2	990	12
19	2	0	2	1050	12
20	2	-1	2	1080	11
n	20	20	20	20	20
Mean	2.20	-0.65	2.10	1049.00	11.90
Variance	0.695	0.871	0.095	9556.842	0.621
s.d.	0.834	0.933	0.308	97.759	0.788
s.e.m.	0.186	0.209	0.069	21.860	0.176

B.carinata parental population TAMU TEXEL GREENS

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	2	1000	11
2	3	-1	2	980	12
3	4	-2	2	1050	12
4	3	-4	2	900	12
5	1	0	2	1150	13
6	4	-2	2	990	12
7	2	-1	3	1100	11
8	3	-2	2	1060	12
9	3	-3	2	990	11
10	2	-1	2	1080	13
11	2	-1	2	980	12
12	1	0	2	1090	11
13	1	0	2	990	12
14	2	-1	3	1450	14
15	2	-2	2	1020	12
16	3	-2	2	740	12
17	2	-1	2	1100	11
18	4	-3	2	890	12
19	3	-2	2	1020	12
20	1	0	2	1100	11
n	20	20	20	20	20
Mean	2.40	-1.40	2.10	1034.00	11.90
Variance	0.989	1.305	0.095	18109.474	0.621
s.d.	0.995	1.142	0.308	134.571	0.788
s.e.m.	0.222	0.255	0.069	30.091	0.176

C3.2 Brassica carinata F1 populations

B.carinata population ADD(CDZ), Generation F1

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	2	1105	11 *
2	2	0	2	1080	12
3	1	0	2	1045	11
4	2	0	2	1110	11
5	2	0	2	1140	11
n	5	5	5	5	5
Mean	1.80	0.00	2.00	1096.00	11.20
Variance	0.200	0.000	0.000	1267.500	0.200
s.d.	0.447	0.000	0.000	35.602	0.447
s.e.m.	0.200	0.000	0.000	15.922	0.200

B.carinata population CDZ(ADD), Generation F1

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	2	1050	10
2	3	-2	2	1105	11
3	2	0	2	1065	11
4	2	0	2	1010	10 *
5	2	0	2	1040	11
n	5	5	5	5	5
Mean	2.20	-0.40	2.00	1054.00	10.60
Variance	0.200	0.800	0.000	1217.500	0.300
s.d.	0.447	0.894	0.000	34.893	0.548
s.e.m.	0.200	0.400	0.000	15.604	0.245

B.carinata population ADD(TTG), Generation F1

Plant	DAD	ADL	StT	L1-F	N1-F
1	3	-2	2	1120	12 *
2	3	-1	1	1100	11
3	2	0	2	1225	12
4	3	-1	2	1205	12
5	2	0	2	1170	12
n	5	5	5	5	5
Mean	2.60	-0.80	1.80	1164.00	11.80
Variance	0.300	0.700	0.200	2867.500	0.200
s.d.	0.548	0.837	0.447	53.549	0.447
s.e.m.	0.245	0.374	0.200	23.948	0.200

B.carinata population TTG(ADD), Generation F1

Plant	DAD	ADL	StT	L1-F	N1-F
1	3	-1	2	1155	12
2	1	0	2	1120	12
3	3	-2	2	1005	11 *
4	2	0	3	1300	13
5	3	-1	2	1210	11
n	5	5	5	5	5
Mean	2.40	-0.80	2.20	1158.00	11.80
Variance	0.800	0.700	0.200	11932.500	0.700
s.d.	0.894	0.837	0.447	109.236	0.837
s.e.m.	0.400	0.374	0.200	48.852	0.374

B.carinata population CDZ(TTG), Generation F1

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	2	1190	12
2	3	-1	1	1080	10
3	4	-2	2	1165	10
4	3	-1	3	1210	12
5	3	-1	2	1145	11 *
n	5	5	5	5	5
Mean	3.00	-1.00	2.00	1158.00	11.00
Variance	0.500	0.500	0.500	2507.500	1.000
s.d.	0.707	0.707	0.707	50.075	1.000
s.e.m.	0.316	0.316	0.316	22.394	0.447

B.carinata population TTG(CDZ), Generation F1

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	2	1335	11
2	2	0	2	1290	11
3	2	0	1	1090	11
4	3	-2	2	1185	10 *
5	2	0	2	1195	10
n	5	5	5	5	5
Mean	2.20	-0.40	1.80	1219.00	10.60
Variance	0.200	0.800	0.200	9217.500	0.300
s.d.	0.447	0.894	0.447	96.008	0.548
s.e.m.	0.200	0.400	0.200	42.936	0.245

C3.3 Brassica carinata crosses ADD x TTG, Generation F2

B.carinata population ADD(TTG), Generation F2

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	2	1005	12
2	3	0	3	825	11 *
3	3	-2	2	1020	12
4	1	0	2	1000	12
5	3	-1	1	910	13
6	2	0	2	895	12 *
7	2	0	2	1160	11
8	2	0	2	1070	13
9	2	0	3	975	13
10	2	0	2	1125	12
11	1	0	2	1180	14
12	1	0	2	1170	12
13	2	0	2	1115	12
14	2	0	2	825	11 *
15	3	-1	1	355	9
n	15	15	15	15	15
Mean	2.07	-0.27	2.00	975.33	11.93
Variance	0.495	0.352	0.286	43498.095	1.352
s.d.	0.704	0.594	0.535	208.562	1.163
s.e.m.	0.182	0.153	0.138	53.850	0.300
Mean(S)				848.33	11.33

B.carinata population TTG(ADD), Generation F2

Plant	DAD	ADL	StT	L1-F	N1-F
1	3	0	2	815	11 *
2	2	0	1	890	11
3	3	0	2	900	10
4	2	0	2	890	12 *
5	1	0	3	1105	14
6	3	0	3	1130	13
7	1	0	2	1000	12
8	3	-3	3	1005	13
9	2	0	3	1150	14
10	3	-2	3	1120	15
11	2	0	3	1305	16
12	2	0	2	1160	14
13	3	2	3	1140	14
14	2	0	2	755	11 *
15	2	0	3	1200	12
n	15	15	15	15	15
Mean	2.27	-0.20	2.47	1037.67	12.80
Variance	0.495	1.171	0.410	25017.381	2.886
s.d.	0.704	1.082	0.640	158.169	1.699
s.e.m.	0.182	0.279	0.165	40.839	0.439
Mean(S)				820.00	11.33

C3.4 Brassica carinata crosses ADD x TTG, Generation F3

B.carinata population ADD(TTG), Generation F3

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	1430	11
2	1	0	2	900	8 *
3	1	0	2	1300	11
4	1	0	2	980	12 *
5	1	0	2	1120	12
6	1	0	2	1145	12
7	1	0	2	1150	9
8	1	0	2	990	9 *
9	1	0	2	990	15
10	1	0	2	1310	14
11	1	0	2	1155	13
12	2	0	2	1180	11
13	1	0	2	1160	13
14	1	0	2	1155	11
15	1	0	2	1210	13
16	1	0	2	980	11 *
17	1	0	2	1145	13
18	1	0	2	1160	14
19	1	0	2	1140	11
20	1	0	2	1020	12
n	20	20	20	20	20
Mean	1.05	0.00	2.00	1131.00	11.75
Variance	0.050	0.000	0.000	16498.947	3.145
s.d.	0.224	0.000	0.000	128.448	1.773
s.e.m.	0.050	0.000	0.000	28.722	0.397
Mean(S)				962.50	10.00

B.carinata population TTG(ADD), Generation F3

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	1080	14
2	1	0	2	1150	14
3	2	0	3	1035	13
4	1	0	2	1060	14
5	1	0	2	1020	15
6	1	0	2	1030	13
7	1	0	2	860	12 *
8	1	0	2	820	10 *
9	1	0	2	1000	13
10	1	0	2	1150	15
11	1	0	2	1120	16
12	1	0	2	1030	13
13	1	0	2	1090	15
14	1	0	2	1070	16
15	1	0	2	1010	15
16	1	0	2	1240	15
17	1	0	2	1160	16
18	1	0	2	950	15 *
19	1	0	2	980	15 *
20	1	0	2	1010	15
n	20	20	20	20	20
Mean	1.05	0.00	2.05	1043.25	14.20
Variance	0.050	0.000	0.050	9827.039	2.274
s.d.	0.224	0.000	0.224	99.131	1.508
s.e.m.	0.050	0.000	0.050	22.166	0.337
Mean(S)				902.50	13.00

C3.5 Brassica carinata crosses ADD x TTG, Generation F4

B.carinata population ADD(TTG), Generation F4

Plant	DAD	ADL	StT	L1-F	N1-F	
1	1	0	2	550	10	*
2	1	0	2	650	10	
3	1	0	3	635	11	
4	1	0	2	740	10	
5	1	0	2	865	12	
6	1	0	2	550	9	*
7	1	0	2	770	10	
8	1	0	1	620	9	
9	1	0	2	590	10	*
10	1	0	2	755	11	
11	1	0	2	825	10	
12	1	0	2	680	9	
13	1	0	2	840	9	
14	1	0	2	760	10	
15	1	0	2	715	9	
16	1	0	2	785	10	
17	1	0	2	690	10	
18	1	0	2	760	11	
19	1	0	2	630	10	*
20	1	0	2	810	11	
n	20	20	20	20	20	
Mean	1.00	0.00	2.00	711.00	10.05	
Variance	0.000	0.000	0.105	9048.947	0.682	
s.d.	0.000	0.000	0.324	95.126	0.826	
s.e.m.	0.000	0.000	0.073	21.271	0.185	
S.Mean				580.00	9.75	

B.carinata population TTG(ADD), Generation F4

Plant	DAD	ADL	StT	L1-F	N1-F	
1	1	0	2	690	11	
2	1	0	2	740	11	
3	2	0	2	660	12	*
4	1	0	2	780	10	
5	1	0	2	685	11	
6	1	0	2	800	11	
7	1	0	2	750	10	
8	1	0	2	710	10	
9	1	0	2	780	10	
10	1	0	2	630	11	*
11	1	0	3	705	10	
12	1	0	2	740	11	
13	1	0	2	730	12	
14	2	0	2	725	13	
15	1	0	2	660	11	
16	1	0	2	610	11	*
17	1	0	3	520	11	*
18	1	0	2	720	11	
19	1	0	2	775	10	
20	1	0	2	685	11	
n	20	20	20	20	20	
Mean	1.10	0.00	2.10	704.75	10.90	
Variance	0.095	0.000	0.095	4467.039	0.621	
s.d.	0.308	0.000	0.308	66.836	0.788	
s.e.m.	0.069	0.000	0.069	14.945	0.176	
S.Mean				605.00	11.25	

C3.6 Brassica carinata crosses CDZ x TTG, Generation F2

B.carinata population CDZ(TTG), Generation F2

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	3	1155	11
2	2	0	2	965	15
3	2	0	3	1200	11
4	2	0	2	1440	13
5	2	0	2	735	11 *
6	1	0	2	1205	11
7	2	0	2	875	11 *
8	2	0	1	810	9
9	3	-1	1	590	11
10	2	0	1	1005	9
11	2	0	1	945	10
12	2	0	1	875	10
13	3	-1	2	710	11 *
14	4	-1	2	645	9
15	2	0	1	905	10
n	15	15	15	15	15
Mean	2.20	-0.20	1.73	937.33	10.80
Variance	0.457	0.171	0.495	54988.810	2.457
s.d.	0.676	0.414	0.704	234.497	1.568
s.e.m.	0.175	0.107	0.182	60.547	0.405
Mean(S)				773.33	11.00

B.carinata population TTG(CDZ), Generation F2

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	2	905	9 *
2	3	0	2	1185	12
3	2	0	2	970	9
4	2	0	2	1490	13
5	2	0	2	1200	11
6	1	0	1	970	10
7	1	0	2	935	10
8	2	0	2	780	10 *
9	2	0	2	860	10 *
10	2	0	1	915	10
11	2	0	2	905	11
12	1	0	3	1315	12
13	2	0	2	1210	11
14	2	0	2	1295	10
15	2	0	2	1200	10
n	15	15	15	15	15
Mean	1.87	0.00	1.93	1075.67	10.53
Variance	0.267	0.000	0.210	42720.952	1.267
s.d.	0.516	0.000	0.458	206.690	1.125
s.e.m.	0.133	0.000	0.118	53.367	0.291
Mean(S)				848.33	9.67

C3.7 Brassica carinata crosses CDZ x TTG, Generation F3

B.carinata population CDZ(TTG), Generation F3

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	670	12 *
2	1	0	2	610	10 *
3	2	0	2	750	12
4	2	0	2	575	12 *
5	1	0	2	890	8
6	1	0	2	1070	9
7	2	0	2	680	10
8	1	0	2	850	9
9	2	0	2	715	13
10	1	0	3	750	12
11	1	0	2	830	8
12	1	0	2	910	11
13	1	0	2	710	9
14	2	0	2	685	9
15	2	0	3	680	12
16	2	0	2	620	10 *
17	2	0	2	860	11
18	2	0	2	825	11
19	2	0	2	705	13
20	1	0	2	860	8
n	20	20	20	20	20
Mean	1.50	0.00	2.10	762.25	10.45
Variance	0.263	0.000	0.095	14906.513	2.787
s.d.	0.513	0.000	0.308	122.092	1.669
s.e.m.	0.115	0.000	0.069	27.301	0.373
Mean(S)				618.75	11.00

B.carinata population TTG(CDZ), Generation F3

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	3	1490	14
2	1	0	1	970	9
3	1	0	1	820	9
4	1	0	1	850	10
5	1	0	3	1165	13
6	1	0	1	670	9
7	1	0	1	1215	11
8	1	0	3	1460	12
9	2	0	2	740	9 *
10	2	0	3	985	9
11	2	0	2	915	10 *
12	1	0	2	1500	14
13	1	0	2	810	7 *
14	1	0	1	770	11
15	1	0	1	1150	12
16	1	0	1	950	10
17	1	0	2	1185	12
18	1	0	2	1200	12
19	1	0	1	785	9
20	2	0	2	745	8 *
n	20	20	20	20	20
Mean	1.20	0.00	1.75	1018.75	10.50
Variance	0.168	0.000	0.618	68807.566	3.842
s.d.	0.410	0.000	0.786	262.312	1.960
s.e.m.	0.092	0.000	0.176	58.655	0.438
Mean(S)				802.50	8.50

C3.8 Brassica carinata crosses CDZ x TTG, Generation F4

B.carinata population CDZ(TTG), Generation F4

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	1	580	9
2	2	0	2	630	10
3	1	0	2	560	10
4	1	0	2	450	9 *
5	2	0	3	600	11
6	1	0	2	655	10
7	2	0	2	580	11
8	2	0	2	600	11
9	1	0	2	570	10
10	1	0	2	475	9 *
11	1	0	2	570	11
12	1	0	2	560	10
13	1	0	2	565	11
14	2	0	2	580	11
15	2	0	2	630	10
16	1	0	2	545	10 *
17	1	0	2	565	9
18	1	0	2	405	10 *
19	1	0	2	815	10
20	1	0	2	805	10
n	20	20	20	20	20
Mean	1.30	0.00	2.00	587.00	10.10
Variance	0.221	0.000	0.105	9343.158	0.516
s.d.	0.470	0.000	0.324	96.660	0.718
s.e.m.	0.105	0.000	0.073	21.614	0.161
Mean(S)				468.75	9.50

B.carinata population TTG(CDZ), Generation F4

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	550	9
2	1	0	2	520	8 *
3	1	0	2	590	8
4	1	0	2	710	8
5	1	0	2	675	8
6	1	0	2	610	8
7	1	0	2	555	9
8	1	0	2	690	9
9	1	0	2	725	9
10	1	0	2	690	8
11	1	0	2	560	7
12	1	0	2	510	10 *
13	1	0	2	540	7 *
14	1	0	2	685	8
15	1	0	2	570	8
16	1	0	2	600	7
17	1	0	2	680	8
18	1	0	2	690	8
19	1	0	2	525	8 *
20	1	0	2	740	8
n	20	20	20	20	20
Mean	1.00	0.00	2.00	620.75	8.15
Variance	0.000	0.000	0.000	5958.618	0.555
s.d.	0.000	0.000	0.000	77.192	0.745
s.e.m.	0.000	0.000	0.000	17.261	0.167
Mean(S)				523.75	8.25

C3.9 Brassica juncea parental populations

B.juncea breeding line J/1078/5/4/89/2013

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	1020	10
2	1	0	2	860	9
3	1	0	2	920	10
4	1	0	3	1060	11
5	1	0	2	1095	11
6	2	0	2	1000	11
7	1	0	2	930	10
8	1	0	2	1245	11
9	1	0	2	1155	11
10	1	0	2	880	10
11	1	0	2	1215	11
12	2	0	2	980	11
13	2	0	2	1110	11
14	1	0	2	1215	11
15	1	0	3	1180	12
16	1	0	2	1120	10
17	1	0	2	995	11
18	1	0	2	1005	11
19	1	0	2	1200	10
20	1	0	2	1185	11
n	20	20	20	20	20
Mean	1.15	0.00	2.10	1068.50	10.65
Variance	0.134	0.000	0.095	14450.263	0.450
s.d.	0.366	0.000	0.308	120.209	0.671
s.e.m.	0.082	0.000	0.069	26.880	0.150

B.juncea variety STOKE

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	680	9
2	1	0	2	720	9
3	1	0	2	1080	10
4	1	0	2	1115	10
5	1	0	2	1050	9
6	2	0	3	1230	10
7	1	0	2	990	10
8	1	0	2	1205	9
9	2	0	2	840	11
10	2	0	2	995	11
11	1	0	2	1200	10
12	1	0	2	1040	9
13	1	0	2	1250	10
14	1	0	2	1095	10
15	1	0	2	1040	11
16	2	0	2	1110	11
17	1	0	3	1210	10
18	1	0	2	1000	9
19	1	0	2	1045	9
20	1	0	2	1100	10
n	20	20	20	20	20
Mean	1.20	0.00	2.10	1049.75	9.85
Variance	0.168	0.000	0.095	24053.882	0.555
s.d.	0.410	0.000	0.308	155.093	0.745
s.e.m.	0.092	0.000	0.069	34.680	0.167

B.juncea variety TROWSE

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	1020	11
2	1	0	2	990	10
3	2	0	2	715	9
4	1	0	2	1200	10
5	2	0	2	880	10
6	2	0	2	1090	10
7	1	0	2	1110	11
8	2	0	3	1180	10
9	2	0	2	1025	10
10	1	0	2	1025	10
11	2	0	2	1210	10
12	3	0	2	1005	10
13	1	0	2	1000	9
14	2	0	2	1080	10
15	2	0	2	1100	10
16	1	0	2	995	9
17	1	0	2	1040	9
18	1	0	2	1105	10
19	2	0	2	1130	11
20	1	0	2	1005	10
n	20	20	20	20	20
Mean	1.55	0.00	2.05	1045.25	9.95
Variance	0.366	0.000	0.050	12585.461	0.366
s.d.	0.605	0.000	0.224	112.185	0.605
s.e.m.	0.135	0.000	0.050	25.085	0.135

C3.10 Brassica juncea F1 populations

B.juncea population J78(STO), Generation F1

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	2	1015	11
2	1	0	2	985	11
3	1	0	2	980	11
4	2	0	2	1000	11
5	2	0	2	870	11 *
n	5	5	5	5	5
Mean	1.60	0.00	2.00	970.00	11.00
Variance	0.300	0.000	0.000	3312.500	0.000
s.d.	0.548	0.000	0.000	57.554	0.000
s.e.m.	0.245	0.000	0.000	25.739	0.000

B.juncea population STO(J78), Generation F1

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	2	870	10
2	1	0	2	925	11
3	2	0	3	950	11
4	2	0	2	840	10 *
5	1	0	2	900	11
n	5	5	5	5	5
Mean	1.60	0.00	2.20	897.00	10.60
Variance	0.300	0.000	0.200	1895.000	0.300
s.d.	0.548	0.000	0.447	43.532	0.548
s.e.m.	0.245	0.000	0.200	19.468	0.245

B.juncea population J78(TRO), Generation F1

Plant	DAD	ADL	StT	L1-F	N1-F
1	3	-1	2	805	10 *
2	2	0	2	890	9
3	1	0	2	910	10
4	2	0	2	950	10
5	2	0	2	905	9
n	5	5	5	5	5
Mean	2.00	-0.20	2.00	892.00	9.60
Variance	0.500	0.200	0.000	2857.500	0.300
s.d.	0.707	0.447	0.000	53.456	0.548
s.e.m.	0.316	0.200	0.000	23.906	0.245

B.juncea population TRO(J78), Generation F1

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	945	10
2	2	0	1	895	10
3	2	0	2	930	10
4	2	0	2	910	11
5	2	0	2	880	10 *
n	5	5	5	5	5
Mean	1.80	0.00	1.80	912.00	10.20
Variance	0.200	0.000	0.200	682.500	0.200
s.d.	0.447	0.000	0.447	26.125	0.447
s.e.m.	0.200	0.000	0.200	11.683	0.200

B.juncea population STO(TRO), Generation F1

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	1075	11
2	2	0	2	1100	11
3	2	0	3	1140	10
4	1	0	2	995	11 *
5	2	0	2	1035	10
n	5	5	5	5	5
Mean	1.60	0.00	2.20	1069.00	10.60
Variance	0.300	0.000	0.200	3167.500	0.300
s.d.	0.548	0.000	0.447	56.281	0.548
s.e.m.	0.245	0.000	0.200	25.169	0.245

B.juncea population TRO(STO), Generation F1

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	990	10
2	2	0	2	945	9
3	2	0	2	1010	10
4	1	0	2	890	10 *
5	3	-1	2	975	10
n	5	5	5	5	5
Mean	1.80	-0.20	2.00	962.00	9.80
Variance	0.700	0.200	0.000	2182.500	0.200
s.d.	0.837	0.447	0.000	46.717	0.447
s.e.m.	0.374	0.200	0.000	20.893	0.200

C3.11 Brassica juncea crosses J78 x STO, Generation F2

B.juncea population J78(STO), Generation F2

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	3	990	11
2	4	-2	3	910	11
3	2	0	3	930	11
4	1	0	2	940	11
5	1	0	2	890	10
6	1	0	2	1220	13
7	1	0	3	1080	12
8	1	0	2	950	14
9	1	0	2	780	10 *
10	1	0	2	790	11
11	1	0	2	670	11 *
12	1	0	2	775	11 *
13	1	0	1	585	9
14	1	0	3	1155	14
15	1	0	1	590	8
n	15	15	15	15	15
Mean	1.33	-0.13	2.20	883.67	11.13
Variance	0.667	0.267	0.457	35676.667	2.695
s.d.	0.816	0.516	0.676	188.883	1.642
s.e.m.	0.211	0.133	0.175	48.769	0.424
Mean(S)				741.67	10.67

B.juncea population STO(J78), Generation F2

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	395	7 *
2	1	0	3	940	10
3	1	0	2	720	12
4	1	0	3	610	10 *
5	1	0	3	915	12
6	1	0	2	705	10
7	1	0	2	600	9 *
8	1	0	2	715	10
9	1	0	2	635	10
10	1	0	2	720	11
11	1	0	1	600	9
12	1	0	3	970	12
13	1	0	3	900	10
14	1	0	2	750	9
15	2	0	1	725	9
n	15	15	15	15	15
Mean	1.07	0.00	2.20	726.67	10.00
Variance	0.067	0.000	0.457	23970.238	1.857
s.d.	0.258	0.000	0.676	154.823	1.363
s.e.m.	0.067	0.000	0.175	39.975	0.352
Mean(S)				535.00	8.67

C3.12 Brassica juncea crosses J78 x STO, Generation F3

B.juncea population J78(STO), Generation F3

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	935	11
2	1	0	1	1000	10
3	1	0	1	990	13
4	1	0	2	1085	15
5	1	0	2	1140	12
6	2	0	3	970	14
7	2	0	3	820	14 *
8	2	0	3	1355	17
9	1	0	1	785	8
10	1	0	3	810	14 *
11	4	-3	3	875	7
12	1	0	2	1295	17
13	1	0	2	1050	12
14	1	0	1	310	7
15	2	0	3	970	14
16	1	0	2	880	15 *
17	2	0	3	1050	16
18	2	0	3	860	14 *
19	1	0	2	1415	15
20	1	0	1	640	8
n	20	20	20	20	20
Mean	1.45	-0.15	2.15	961.75	12.65
Variance	0.576	0.450	0.661	61263.882	10.134
s.d.	0.759	0.671	0.813	247.515	3.183
s.e.m.	0.170	0.150	0.182	55.346	0.712
Mean(S)				842.50	14.25

B.juncea population STO(J78), Generation F3

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	2	850	12
2	1	0	2	810	9 *
3	1	0	1	825	10
4	1	0	1	940	10
5	1	0	1	720	11
6	1	0	2	830	11 *
7	1	0	2	1015	11
8	1	0	2	1035	12
9	1	0	2	1040	13
10	1	0	1	730	7
11	1	0	3	845	12
12	2	0	2	790	12 *
13	1	0	2	810	11 *
14	1	0	2	835	11
15	1	0	3	1270	15
16	2	0	3	1100	14
17	2	0	2	850	13
18	2	0	2	1235	14
19	1	0	2	855	11
20	1	0	1	710	10
n	20	20	20	20	20
Mean	1.25	0.00	1.90	904.75	11.45
Variance	0.197	0.000	0.411	25885.461	3.418
s.d.	0.444	0.000	0.641	160.890	1.849
s.e.m.	0.099	0.000	0.143	35.976	0.413
Mean(S)				810.00	10.75

C3.13 Brassica juncea crosses J78 x STO, Generation F4

B.juncea population J78(STO), Generation F4

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	2	565	8
2	1	0	2	500	7 *
3	2	0	2	630	8
4	2	0	3	590	9
5	3	0	2	520	9
6	1	0	2	580	8
7	1	0	2	535	8
8	2	0	2	480	8 *
9	2	0	2	535	9
10	2	0	2	550	9
11	2	0	2	650	8
12	2	0	2	475	8 *
13	1	0	2	470	6 *
14	2	0	2	605	9
15	2	0	2	560	8
16	2	0	2	530	9
17	2	0	2	535	8
18	2	0	2	590	8
19	2	0	2	615	8
20	2	0	2	585	7
n	20	20	20	20	20
Mean	1.85	0.00	2.05	555.00	8.10
Variance	0.239	0.000	0.050	2642.105	0.621
s.d.	0.489	0.000	0.224	51.401	0.788
s.e.m.	0.109	0.000	0.050	11.494	0.176
Mean(S)				481.25	7.25

B.juncea population STO(J78), Generation F4

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	475	9
2	2	0	2	505	8
3	2	0	2	430	6 *
4	2	0	2	580	8
5	2	0	2	580	10
6	2	0	2	470	8 *
7	2	0	2	450	9 *
8	1	0	2	480	7
9	2	0	2	505	8
10	2	0	2	560	7
11	2	0	2	500	7
12	2	0	2	470	7
13	2	0	2	560	7
14	2	0	1	500	7
15	2	0	2	450	6 *
16	1	0	2	535	7
17	2	0	2	520	8
18	2	0	2	485	9
19	4	0	2	430	8
20	2	0	2	470	8
n	20	20	20	20	20
Mean	1.95	0.00	1.95	497.75	7.70
Variance	0.366	0.000	0.050	2122.303	1.063
s.d.	0.605	0.000	0.224	46.068	1.031
s.e.m.	0.135	0.000	0.050	10.301	0.231
Mean(S)				450.00	7.25

C3.14 Brassica juncea crosses J78 x TRO, Generation F2

B.juncea population J78(TRO), Generation F2

Plant	DAD	ADL	StT	L1-F	N1-F	
1	2	0	2	580	8	*
2	2	0	2	680	9	
3	1	0	1	700	9	
4	2	0	2	760	9	
5	2	0	2	670	8	*
6	2	0	3	760	9	
7	2	0	1	830	9	
8	3	-1	2	800	10	
9	1	0	2	810	10	
10	3	0	1	820	10	
11	1	0	2	650	8	*
12	-	0	2	790	8	
13	1	0	1	740	9	
14	2	0	3	840	8	
15	2	0	1	755	8	
n	15	15	15	15	15	
Mean	1.87	-0.07	1.80	745.67	8.80	
Variance	0.410	0.067	0.457	5703.095	0.600	
s.d.	0.640	0.258	0.676	75.519	0.775	
s.e.m.	0.165	0.067	0.175	19.499	0.200	
Mean(S)				633.33	8.00	

B.juncea population TRO(J78), Generation F2

Plant	DAD	ADL	StT	L1-F	N1-F	
1	2	0	2	890	10	*
2	1	0	1	880	11	
3	3	0	3	900	10	
4	1	0	2	870	10	*
5	1	0	1	810	10	
6	2	0	2	880	11	*
7	3	-1	1	740	10	
8	1	0	1	730	10	
9	2	0	2	910	10	
10	1	0	2	900	10	
11	2	0	1	820	10	
12	2	0	2	890	11	
13	2	0	2	900	10	
14	2	0	1	890	9	
15	1	0	1	760	11	
n	15	15	15	15	15	
Mean	1.73	-0.07	1.60	851.33	10.20	
Variance	0.495	0.067	0.400	3940.952	0.314	
s.d.	0.704	0.258	0.632	62.777	0.561	
s.e.m.	0.182	0.067	0.163	16.209	0.145	
Mean(S)				880.00	10.33	

C3.15 Brassica juncea crosses J78 x TRO, Generation F3

B.juncea population J78(TRO), Generation F3

Plant	DAD	ADL	StT	L1-F	N1-F	
1	1	0	2	1180	11	
2	1	0	3	1300	11	
3	2	0	3	800	10	*
4	1	0	2	1000	10	
5	1	0	2	985	10	
6	1	0	2	740	10	*
7	2	0	3	880	11	
8	1	0	2	680	11	*
9	1	0	2	825	11	*
10	1	0	2	825	11	
11	2	0	3	1495	11	
12	1	0	1	490	9	
13	1	0	1	655	8	
14	1	0	2	915	10	
15	1	0	1	640	9	
16	1	0	1	600	8	
17	1	0	2	1115	10	
18	1	0	2	1020	10	
19	1	0	1	985	9	
20	1	0	2	1095	11	
n	20	20	20	20	20	
Mean	1.15	0.00	1.95	911.25	10.05	
Variance	0.134	0.000	0.471	62873.355	0.997	
s.d.	0.366	0.000	0.686	250.746	0.999	
s.e.m.	0.082	0.000	0.153	56.068	0.223	
Mean(S)				761.25	10.50	

B.juncea population TRO(J78), Generation F3

Plant	DAD	ADL	StT	L1-F	N1-F	
1	1	0	3	910	12	
2	5	-10	2	805	12	
3	2	0	2	750	12	*
4	1	0	2	960	11	
5	1	0	2	820	11	
6	1	0	3	825	11	
7	1	0	2	755	12	
8	2	0	3	840	11	
9	2	0	3	855	12	
10	1	0	3	910	11	
11	2	0	3	820	11	
12	1	0	3	580	12	*
13	2	0	3	560	11	*
14	1	0	3	930	13	
15	1	0	3	625	12	*
16	2	0	3	885	11	
17	1	0	2	815	11	
18	1	0	2	820	12	
19	2	0	2	800	12	
20	1	0	3	790	11	
n	20	20	20	20	20	
Mean	1.55	-0.50	2.60	802.75	11.55	
Variance	0.892	5.000	0.253	11669.671	0.366	
s.d.	0.945	2.236	0.503	108.026	0.605	
s.e.m.	0.211	0.500	0.112	24.155	0.135	
Mean(S)				628.75	11.75	

C3.16 Brassica juncea crosses J78 x TRO, Generation F4

B.juncea population J78(TRO), Generation F4

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	475	8
2	1	0	2	515	7
3	1	0	2	530	8
4	1	0	2	510	7
5	1	0	2	400	7 *
6	1	0	2	395	8 *
7	1	0	2	560	7
8	1	0	1	430	8
9	1	0	1	385	8
10	1	0	2	560	7
11	1	0	2	565	8
12	1	0	2	495	7
13	1	0	2	470	8
14	1	0	2	485	8
15	1	0	1	400	7
16	1	0	2	535	8
17	1	0	2	500	8
18	1	0	2	510	8
19	1	0	2	415	8 *
20	1	0	2	385	8 *
n	20	20	20	20	20
Mean	1.00	0.00	1.85	476.00	7.75
Variance	0.000	0.000	0.134	3885.789	0.303
s.d.	0.000	0.000	0.366	62.336	0.550
s.e.m.	0.000	0.000	0.082	13.939	0.123
Mean(S)				398.75	7.75

B.juncea population TRO(J78), Generation F4

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	460	8
2	1	0	1	470	8
3	1	0	2	395	6 *
4	4	-4	2	420	9
5	1	0	2	560	8
6	1	0	2	445	7 *
7	1	0	2	500	8
8	1	0	2	565	7
9	1	0	2	560	8
10	1	0	1	430	7
11	1	0	2	450	6
12	1	0	2	555	8
13	1	0	3	440	7 *
14	1	0	2	520	8
15	1	0	2	460	7
16	1	0	2	470	7
17	1	0	2	450	7
18	1	0	2	435	6 *
19	1	0	2	470	8
20	1	0	2	570	7
n	20	20	20	20	20
Mean	1.15	-0.20	1.95	481.25	7.35
Variance	0.450	0.800	0.155	2989.145	0.661
s.d.	0.671	0.894	0.394	54.673	0.813
s.e.m.	0.150	0.200	0.088	12.225	0.182
Mean(S)				428.75	6.50

C3.17 Brassica juncea crosses STO x TRO, Generation F2

B.juncea population STO(TRO), Generation F2

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	2	700	8 *
2	2	0	2	910	8
3	1	0	2	760	10
4	2	0	2	635	11 *
5	2	0	2	875	9
6	2	0	3	960	13
7	2	0	2	745	8
8	2	0	2	890	10
9	2	0	2	905	9
10	2	0	2	810	11
11	3	-1	2	795	10
12	2	0	1	780	9
13	2	0	2	790	11
14	1	0	2	845	11
15	2	0	2	730	9 *
n	15	15	15	15	15
Mean	1.93	-0.07	2.00	808.67	9.80
Variance	0.210	0.067	0.143	7908.810	2.029
s.d.	0.458	0.258	0.378	88.931	1.424
s.e.m.	0.118	0.067	0.098	22.962	0.368
Mean(S)				688.33	9.33

B.juncea population TRO(STO), Generation F2

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	3	735	9 *
2	2	0	2	860	9
3	1	0	2	700	9 *
4	2	0	2	615	10 *
5	2	0	2	900	9
6	2	0	3	945	9
7	4	-1	3	810	8
8	3	0	2	1000	10
9	1	0	2	995	9
10	3	-1	2	860	9
11	3	-1	2	885	10
12	1	0	1	810	10
13	3	-1	1	915	11
14	2	0	2	985	11
15	2	0	1	970	10
n	15	15	15	15	15
Mean	2.13	-0.27	2.00	865.67	9.53
Variance	0.838	0.210	0.429	13128.095	0.695
s.d.	0.915	0.458	0.655	114.578	0.834
s.e.m.	0.236	0.118	0.169	29.584	0.215
Mean(S)				683.33	9.33

C3.18 Brassica juncea crosses STO x TRO, Generation F3

B.juncea population STO(TRO), Generation F3

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	1	710	12
2	1	0	2	945	13
3	1	0	1	820	11
4	1	0	1	540	10
5	2	0	3	1290	14
6	1	0	1	710	11
7	2	0	2	810	12 *
8	1	0	2	785	12 *
9	1	0	2	1080	15
10	1	0	2	715	10 *
11	2	0	2	825	12 *
12	1	0	2	885	11
13	2	0	2	830	12
14	2	0	2	920	13
15	1	0	2	990	12
16	2	0	2	875	13
17	2	0	3	1155	18
18	2	0	2	995	13
19	1	0	2	1175	13
20	1	0	3	1310	15
n	20	20	20	20	20
Mean	1.40	0.00	1.95	918.25	12.60
Variance	0.253	0.000	0.366	41119.145	3.516
s.d.	0.503	0.000	0.605	202.779	1.875
s.e.m.	0.112	0.000	0.135	45.343	0.419
Mean(S)				783.75	11.50

B.juncea population TRO(STO), Generation F3

Plant	DAD	ADL	StT	L1-F	N1-F
1	2	0	3	1220	12
2	2	0	3	1330	12
3	1	0	3	1210	10
4	2	0	3	1800	11
5	1	0	3	1405	10
6	1	0	2	1310	12
7	2	0	3	1255	12
8	1	0	3	910	10 *
9	1	0	3	1155	11 *
10	2	0	2	1525	12
11	1	0	3	1180	12
12	2	0	2	1365	11
13	2	0	2	1720	11
14	1	0	3	1090	11 *
15	1	0	2	1505	11
16	1	0	2	1430	11
17	1	0	3	1180	12
18	2	0	2	1400	10
19	1	0	2	1140	10 *
20	2	0	3	1310	12
n	20	20	20	20	20
Mean	1.45	0.00	2.60	1322.00	11.15
Variance	0.261	0.000	0.253	44251.053	0.661
s.d.	0.510	0.000	0.503	210.359	0.813
s.e.m.	0.114	0.000	0.112	47.038	0.182
Mean(S)				1073.75	10.50

C3.19 Brassica juncea crosses STO x TRO, Generation F4

B.juncea population STO(TRO), Generation F4

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	570	7
2	1	0	2	400	7 *
3	1	0	2	540	8
4	1	0	2	475	7
5	1	0	2	550	8
6	1	0	2	410	6 *
7	1	0	2	485	7
8	1	0	2	520	7
9	1	0	1	415	7
10	1	0	2	540	7
11	1	0	2	440	6
12	1	0	2	360	7 *
13	1	0	2	440	8
14	1	0	2	460	7
15	1	0	2	380	6 *
16	1	0	2	500	7
17	1	0	2	445	7
18	1	0	2	490	8
19	1	0	1	470	6
20	1	0	2	420	8
n	20	20	20	20	20
Mean	1.00	0.00	1.90	465.50	7.05
Variance	0.000	0.000	0.095	3505.000	0.471
s.d.	0.000	0.000	0.308	59.203	0.686
s.e.m.	0.000	0.000	0.069	13.238	0.153
Mean(S)				387.50	6.50

B.juncea population TRO(STO), Generation F4

Plant	DAD	ADL	StT	L1-F	N1-F
1	1	0	2	510	7
2	1	0	2	420	8
3	1	0	2	450	7
4	1	0	2	395	7 *
5	1	0	2	470	7
6	1	0	2	435	7
7	1	0	2	440	6
8	1	0	2	380	7 *
9	1	0	2	410	8 *
10	1	0	2	460	8
11	1	0	2	395	7 *
12	1	0	2	515	8
13	1	0	2	505	8
14	1	0	2	485	7
15	1	0	2	450	8
16	1	0	2	490	7
17	1	0	1	435	7
18	1	0	2	460	9
19	1	0	2	450	6
20	2	0	2	415	6
n	20	20	20	20	20
Mean	1.05	0.00	1.95	448.50	7.25
Variance	0.050	0.000	0.050	1550.263	0.618
s.d.	0.224	0.000	0.224	39.373	0.786
s.e.m.	0.050	0.000	0.050	8.804	0.176
Mean(S)				395.00	7.25

APPENDIX C4

Single-factor analysis of variance: L1-F generation F_1, all breeding lines

Summary

Groups	Count	Sum	Mean	Variance
CC	5	5820	1164	2867.5
CD	5	5790	1158	11932.5

Source of Variation

	SS	df	MS	F	P-value	F crit
Between Groups	90	1	90	0.012162162	0.914901981	5.317644991
Within Groups	59200	8	7400			
Total	59290	9				

Summary

Groups	Count	Sum	Mean	Variance
CH	5	5790	1158	2507.5
CK	5	6095	1219	9217.5

Source of Variation

	SS	df	MS	F	P-value	F crit
Between Groups	9303	1	9302.5	1.586780384	0.243292369	5.317644991
Within Groups	46900	8	5862.5			
Total	56203	9				

Summary

Groups	Count	Sum	Mean	Variance
JA	5	4850	970	3312.5
JB	5	4485	897	1895

Source of Variation

	SS	df	MS	F	P-value	F crit
Between Groups	13322.5	1	13322.5	5.116658665	0.053554672	5.317644991
Within Groups	20830	8	2603.75			
Total	34152.5	9				

Summary

Groups	Count	Sum	Mean	Variance
JC	5	4460	892	2857.5
JD	5	4560	912	682.5

Source of Variation

	SS	df	MS	F	P-value	F crit
Between Groups	1000	1	1000	0.564971751	0.473794139	5.317644991
Within Groups	14160	8	1770			
Total	15160	9				

Summary

Groups	Count	Sum	Mean	Variance
JH	5	5345	1069	3167.5
JK	5	4810	962	2182.5

Source of Variation

	SS	df	MS	F	P-value	F crit
Between Groups	28622.5	1	28622.5	10.7	0.011337806	5.317644991
Within Groups	21400	8	2675			
Total	50022.5	9				

APPENDIX C5

Response of L1-F (expressed as deviation from F1 mean) to selection, and estimates of realised heritability

Brassica carinata

Breeding line		Mean L1-F (selected)	Selection differential *(S)*	Cumulative selection differential *(cS)*	Observed population mean L1-F	Fitted population L1-F	Realised heritability *(h**2)*
CC	1	-60.0	60.0	0.0	0.0	0.0	
	2	-331.7	179.8	60.0	-151.9	-55.4	
	3	-217.5	168.5	239.8	-49.0	-221.5	
	4	-600.0	135.8	408.3	-464.2	-377.1	*0.92*
CD	1	-153.0	153.0	0.0	0.0	0.0	
	2	-338.0	217.7	153.0	-120.3	-106.0	
	3	-255.5	140.8	370.7	-114.7	-256.9	
	4	-553.0	99.8	511.5	-453.2	-354.4	*0.69*
CH	1	-36.7	36.7	0.0	0.0	0.0	
	2	-408.4	262.3	36.7	-146.1	-50.4	
	3	-562.9	143.5	299.0	-419.4	-410.8	
	4	-712.9	118.6	442.5	-594.3	-608.0	*1.37*
CK	1	-66.3	66.3	0.0	0.0	0.0	
	2	-402.9	247.8	66.3	-155.1	-60.6	
	3	-448.8	306.1	314.1	-142.7	-287.3	
	4	-727.5	97.0	620.2	-630.5	-567.3	*0.91*

Brassica juncea

Breeding line		Mean L1-F (selected)	Selection differential *(S)*	Cumulative selection differential *(cS)*	Observed population mean L1-F	Fitted population L1-F	Realised heritability *(h**2)*
JA	1	-100.0	100.0	0.0	0.0	0.0	
	2	-228.3	189.1	100.0	-39.2	-55.5	
	3	-127.5	202.9	289.1	75.4	-160.5	
	4	-488.7	73.7	492.0	-415.0	-273.1	*0.56*
JB	1	-57.0	57.0	0.0	0.0	0.0	
	2	-362.0	201.5	57.0	-160.5	-38.7	
	3	-87.0	134.7	258.5	47.7	-175.3	
	4	-447.0	51.4	393.2	-395.6	-266.7	*0.68*
JC	1	-87.0	87.0	0.0	0.0	0.0	
	2	-258.7	100.7	87.0	-158.0	-65.7	
	3	-130.7	229.0	187.7	98.3	-141.7	
	4	-493.2	89.7	416.7	-403.5	-314.7	*0.76*
JD	1	-36.3	36.3	0.0	0.0	0.0	
	2	-36.3	12.5	36.3	-23.8	-69.3	
	3	-287.5	173.8	48.8	-113.7	-93.1	
	4	-487.5	59.7	222.6	-427.8	-424.9	*1.91*
JH	1	-74.0	74.0	0.0	0.0	0.0	
	2	-380.7	122.4	74.0	-258.3	-103.3	
	3	-285.2	190.3	196.4	-94.9	-274.0	
	4	-681.5	80.6	386.7	-600.9	-539.6	*1.40*
JK	1	-72.0	72.0	0.0	0.0	0.0	
	2	-278.7	178.5	72.0	-100.2	-39.3	
	3	111.8	248.2	250.5	360.0	-136.7	
	4	-567.0	54.2	498.7	-512.8	-272.1	*0.55*

APPENDIX D1

Dictionary of variable names used in SATSUMA

TYPES

Plant_Population_Array: defined TYPE variable, INTEGER ARRAY denoting the 2-dimensional simulated field of plants.

ARRAYS

Plant_at: ARRAY of type PLANT_POPULATION_ARRAY, denoting X,Y co-ordinates of the plant under consideration.

GLOBAL VARIABLES

Chance_of_Secondary_Infection: REAL values 0 to 100, derived from underlying pathogen activity and the potential for secondary infection. Unless this value exceeds a RANDOM deviate (0-99), secondary infections are precluded.

Chance_Susceptible: INTEGER values 50 to 100, "noise" variable giving probability of an infection being successful, expressed as a percentage.

Colour_Display: BOOLEAN, value FALSE will restrict all screen displays to monochrome character set.

Cycle_number: INTEGER, counter incremented with each pass through the program (ie. twice per pathogen life-cycle).

Cycle_Phase: CHARACTER variable (I or M) mutually distinguishing the infection and maturation phases within each Pathogen_Cycle.

Data_Output: TEXT variable used as the logical (device) name for the data output file, to enable generated values to be written to file as opposed to defaulting to the screen.

Data_OutputFile_Name: 1 to 8 character STRING denoting the non-default MS-DOS filename for the output data (.DAT) file.

Dbug_Output: TEXT variable used as the logical (device) name for a storage file used for monitoring/testing during debugging. No function in operational model.

Field_X1: BYTE, component of the top-left position of the Field window.

Field_X2: BYTE, component of the bottom-right position of the Field window.

Field_Y1: BYTE, component of the top-left position of the Field window.

Field_Y2: BYTE, component of the bottom-right position of the Field window.

Max_Number_of_Plants_in_Field: INTEGER, the product of Max_Y and Max_X giving the number of plants in the simulated popuation.

Max_X: INTEGER, denotes the X-dimension of the simulated field.

Max_Y: INTEGER, denotes the Y-dimension of the simulated field.

Mean_Temperature: REAL values 0 to 35, denoting the mean temperature for the duration of the simulation run.

Mean_WindDirection: 1 to 2 character STRING variable (values N, NE, E, SE, S, SW, W and NW) denoting prevailing wind direction for the duration of the simulation run.

Mean_WindSpeed: REAL values 0 to 10, denoting mean windspeed in metres-per-second, for each Pathogen_Cycle.

Message_X1: BYTE, component of the top-left position of the Message window.

Message_X2: BYTE, component of the bottom-right position of the Message window.

Message_Y1: BYTE, component of the top-left position of the Message window.

Message_Y2: BYTE, component of the bottom-right position of the Message window.

Mode_of_Pathogen_Spread: 1 to 9 character STRING variable (values "AIRBORNE", "SPLASH" and "SOILBORNE") denoting the method of pathogen dispersal. Used to control executable progression through the program.

Pathogen_Cycle: INTEGER, counter corresponding to the complete pathogen life-cycle. Incremented with each pass through the pathogen maturation phase, ie. alternate Cycle_Numbers.

Probability_of_Pathogen_Growth: REAL values 0 to 100, denoting the temperature-established underlying pathogen activity.

Proceed_to_Next_Cycle: one-character STRING variable, used in debugging to interrupt program execution. No function in operational model.

Rainfall: 1 to 7 character STRING variable (values "DRY", "TYPICAL" and "WET") denoting the level of rainfall.

Realised_Temperature: unrestricted REAL values (in practice appx. -25 to +60). Temperature per Pathogen_Cycle generated by sampling from the normal distribution specified by Mean_Temperature and Temperature_sd.

Realised_WindDirection: 1 to 2 character STRING variable (values N, NE, E, SE, S, SW, W and NW) denoting actual wind direction (derived from Mean_WindDirection and Wind_Arc) for each Pathogen_Cycle.

Realised_WindSpeed: unrestricted REAL values. Wind speed per Pathogen_Cycle generated by sampling from the normal distribution specified by Mean_WindSpeed and WindSpeed_sd.

Secondary_Infection_Potential: INTEGER values 1, 2 and 4, scaling factor relating potential to initiate secondary infections to underlying pathogen activity.

Secondary_Target: BOOLEAN variable, denotes whether or not the plant in question is positioned such that it is a potential target for secondary infection.

See_Infectious: BOOLEAN variable, value TRUE will display different screen characters for Infectious and Infected plants, FALSE will display both conditions as Infected.

Splash_Factor: INTEGER values 1 to 3, scaling factor, proportional to level of rainfall, which directly determines the infective range of the Primary_Infection operation for splash-dispersed pathogens.

Summary_X1: BYTE, component of the top-left position of the Summary window.

Summary_X2: BYTE, component of the bottom-right position of the Summary window.

Summary_Y1: BYTE, component of the top-left position of the Summary window.

Summary_Y2: BYTE, component of the bottom-right position of the Summary window.

Temperature_Deviation: unrestricted REAL values, used in the uniform to normal transformation procedure to determine a deviation of temperature from the specified mean. Used in intermediate step to calculate Realised_Temperature.

Temperature_sd: REAL values 0 to 10, denoting the standard deviation of temperature about the specified mean.

Total_Dead: INTEGER values 0 to Max_Number_of_Plants_in_Field, denoting number of Dead plants in simulated population.

Total_Healthy: INTEGER values 0 to Max_Number_of_Plants_in_Field, denoting number of Healthy plants in simulated population.

Total_Infected: INTEGER values 0 to Max_Number_of_Plants_in_Field, denoting number of Infected (including Infectious) plants in simulated population.

Total_Infectious: INTEGER values 0 to Max_Number_of_Plants_in_Field, denoting number of Infectious (sub-set of Infected) plants in simulated population.

What_Next: CHARACTER variable (values C,R,S,V and X) used in the Exit_Options FUNCTION as a control switch to specify executable flow through the program.

Wind_Arc: INTEGER values 0, 90, 180, 270 and 360, denoting the arc through which wind direction might vary, centred upon the prevailing wind direction.

WindSpeed_Deviation: unrestricted REAL values, used in the uniform to normal transformation procedure to determine a deviation of wind speed from the specified mean. Used in intermediate step to calculate Realised_WindSpeed.

WindSpeed_Factor: INTEGER values 0 to 3, scaling factor, derived from Realised_WindSpeed, which directly determines the infective range of the Primary_Infection operation for air-borne pathogens.

WindSpeed_sd: REAL values 0 to 10, denoting the standard deviation of wind speed about the specified mean.

WindSpeed_Threshold: REAL values 0 to 10, denoting a threshold wind speed below which infections are precluded for air-borne pathogens.

X_Infection_2: BYTE values 1 to X_Max, denotes Y co-ordinate of potential secondary infection.

X_Plant: BYTE values 1 to X_Max, individual plant X co-ordinate. Component of PLANT_AT Array.

X0: INTEGER value 1 to X_Max, denotes X co-ordinate of primary infection focus.

Y_Infection_2: BYTE values 1 to Y_Max, denotes Y co-ordinate of potential secondary infection.

Y_Plant: BYTE values 1 to Y_Max, individual plant Y co-ordinate. Component of PLANT_AT Array.

Y0: INTEGER value 1 to Y_Max, denotes Y co-ordinate of primary infection focus.

LOCALLY-DECLARED VARIABLES

Answer: locally-declared STRING or CHARACTER variable. User response to screen prompt.

Answer_as_Real: locally-declared REAL variable, used in the conversion of STRING response to REAL number.

Converted_OK: locally-declared INTEGER variable (operates as BOOLEAN) to confirm conversion of STRING value to REAL number.

Counter: locally-declared INTEGER variable commonly used in conjunction with UPCASE FUNCTION to convert all CHARACTER/STRING responses to upper case.

Declared_Focus: locally-declared BOOLEAN variable specifying a manually assigned primary infection focus in preference to a randomly positioned focus.

Declared_Threshold: locally-declared BOOLEAN variable denoting the assignation of a WindSpeed_Threshold value. A returned FALSE value functions as a short-cut in program execution.

Exit_Approval: locally-declared single character STRING variable, which upon Total_Dead equalling or exceeding Max_Number_of_Plants_in_field, exits program to Exit_Options screen upon key press.

Go: locally-declared non-specific CHARACTER variable used in conjunction with the READKEY FUNCTION to effect a non-interactive interruption of program execution.

Response: locally-declared CHARACTER variable operating in the same way as Answer.

RInt99: locally-declared INTEGER variable, used for holding a random value 0 to 99 for subsequent use.

Summary_Approved: 1 to 3 character locally-declared STRING variable. Returned values other than Y or YES prevent program progression and repeat preceding user-interface routines.

U1: locally-declared random REAL deviate, value 0.001 to 1, used in Marsaglia-Bray transformation routine.

U2: locally-declared random REAL deviate, value 0.001 to 1, used in Marsaglia-Bray transformation routine.

U3: locally-declared random REAL deviate, value 0.001 to 1, used in Marsaglia-Bray transformation routine.

V1: locally-declared random REAL deviate, value -1 to 1, used in Marsaglia-Bray transformation routine.

V2: locally-declared random REAL deviate, value -1 to 1, used in Marsaglia-Bray transformation routine.

Variable_WindDirection: locally-declared BOOLEAN variable specifying constant vs. variable wind direction. A returned FALSE value functions as a short-cut in program execution.

X_At_Risk: locally-declared BYTE variable specifying X co-ordinate within a Plant_At address, assessed within Primary_Infection range.

Y_At_Risk: locally-declared BYTE variable specifying Y co-ordinate within a Plant_At address, assessed within Primary_Infection range.

FUNCTIONS

Accept_Integer: FUNCTION operating on a locally specified question to verify and accept a user-specified INTEGER value into the model.

Accept_Real: FUNCTION operating on a locally specified question to verify (where necessary convert an INTEGER value to REAL) and accept a user-specified REAL or INTEGER value into the model.

Ask_Quit: FUNCTION returning a BOOLEAN variable by prompting for a "Q" response to a locally-specified question.

Ask_User: FUNCTION returning a BOOLEAN variable by prompting for a YES/NO response to a locally-specified question.

Discontinue: FUNCTION returning a BOOLEAN variable in response to locally-specified instructions. Discontinue is a complex FUNCTION, making internal reference to the Ask_Quit FUNCTION.

Exit_Options: FUNCTION prompting for a CHARACTER variable functioning to control executable flow through the program.

CONSTANTS

Dead: CONSTANT value 3, one of the 4 plant conditions.
Healthy: CONSTANT value 0, one of the 4 plant conditions.
Infected: CONSTANT value 1, one of the 4 plant conditions.
Infectious: CONSTANT value 2, one of the 4 plant conditions.
Program_Name: CONSTANT, used to specify default output filenames.

APPENDIX D2

SATSUMA Program code

```
{ Comment: SATSUMA ("Spatial And Temporal Simulation Under Mechanistic
  Assumptions") is a mechanistic model, designed as a teaching aid
  to illustrate how intuitively simple patterns of disease spread can
  be described by a logistic mathematical model. The original idea was
  developed by Professor Norman W. Simmonds.}

PROGRAM SATSUMA;

USES Crt, Dos, Graph;

CONST
  Program_Name = 'SATSUMA';
  Healthy = 0;
  Infected = 1;
  Infectious = 2;
  Dead = 3;

{ Comment: The declared "Program Name" is used in the specification of
  output files.
  Plant conditions, HEALTHY, INFECTED, INFECTIOUS (a sub-set of Infected)
  and DEAD are represented in the plant population array as numerical
  values. In theory there is no reason why "Plant Condition" could not be
  declared as a separate type, with the states as alternative values of
  Plant Condition. }

TYPE
  Plant_Population_Array = ARRAY[0..45,0..55] OF INTEGER;

{ Comment: The maximum Y,X values within the Y*X population array are
  40 and 50 respectively. The Plant_Population_Array is deliberately
  "oversized" to provide a rudimentary "event trapping" mechanism. In
  the event of a 40*50 array being specified, "scrolling" is observed:
  an infectious plant at Y=20, X=50 infects its immediate neighbours
  as anticipated, but also initiates a spurious infection at position
  Y=20, X=1. I currently have no explanation for why the array acts in
  a "circular" manner. }

VAR
  Plant_at: PLANT_POPULATION_ARRAY;

  Field_Y1, Field_X1, Field_Y2, Field_X2: BYTE;
  Message_Y1, Message_X1, Message_Y2, Message_X2: BYTE;
  Summary_Y1, Summary_X1, Summary_Y2, Summary_X2: BYTE;
  Y_Plant, X_Plant: BYTE;
  Y_Infection_2, X_Infection_2: BYTE;

  Y0, X0: INTEGER;
  Cycle_number: INTEGER;
  Pathogen_Cycle: INTEGER;
  Total_Healthy: INTEGER;
  Total_Infected: INTEGER;
  Total_Infectious: INTEGER;
  Total_Dead: INTEGER;
  Max_Y, Max_X: INTEGER;
  Max_Number_of_Plants_in_Field: INTEGER;
  Chance_Susceptible: INTEGER;
  Secondary_Infection_Potential: INTEGER;
  Wind_Arc: INTEGER;
  WindSpeed_Factor: INTEGER;
  Splash_Factor: INTEGER;

  Secondary_Target: BOOLEAN;
  See_Infectious: BOOLEAN;
```

```
  Colour_Display: BOOLEAN;

  Chance_of_Secondary_Infection: REAL;
  Mean_Temperature: REAL;
  Realised_Temperature: REAL;
  Temperature_sd: REAL;
  Temperature_Deviation: REAL;
  Probability_of_Pathogen_Growth: REAL;
  WindSpeed_Threshold: REAL;
  Mean_WindSpeed: REAL;
  WindSpeed_Deviation: REAL;
  WindSpeed_sd: REAL;
  Realised_WindSpeed: REAL;

  What_Next: CHAR;
  Proceed_to_Next_Cycle: STRING[1];
  Mode_of_Pathogen_Spread: STRING[9];
  Mean_WindDirection: STRING[2];
  Realised_WindDirection: STRING[2];
  Rainfall: STRING[7];
  Data_OutputFile_Name: STRING[8];
  Cycle_Phase: CHAR;

  Data_Output: TEXT;
  Dbug_Output: TEXT;


{ Comment: The function Accept_Integer operates on a locally specified
  question, prompting the user to enter an integer value. The function
  will only accept the tendered value if it is an integer within the
  specified maximum and minimum values, and in the event that the value
  is unsuitable, repeats the question until an acceptable value is given.}

FUNCTION Accept_Integer
        (Question:STRING; Minimum,Maximum:INTEGER; Default:STRING): INTEGER;
  VAR
    Answer:STRING;            { User response to screen prompt }
    Answer_as_Real:REAL;      { Conversion of STRING response to REAL number }
    Converted_OK:INTEGER;     { Confirmation of conversion to REAL number }
  BEGIN
  REPEAT
    REPEAT
      WRITE(Question);
      READLN(Answer);
      IF ((Answer='') AND (Default<>'')) THEN Answer:=Default;
      VAL(Answer,Answer_as_Real,Converted_OK);
    UNTIL Converted_OK=0
  UNTIL (Answer_as_Real >= Minimum)
    AND (Answer_as_Real <= Maximum)
    AND (Answer_as_Real = INT(Answer_as_Real));
  Accept_Integer:=TRUNC(Answer_as_Real);
  END;


{ Comment: The function Accept_Real operates on a locally specified
  question, prompting the user to enter an real value. The function
  accepts real and integer values between specified maximum and minimum
  values, converting integers to real variables prior to their use within
  the program. In the event that the value is unsuitable, the question
  will be repeated until an acceptable value is tendered.}

FUNCTION Accept_Real
        (Question:STRING; Minimum,Maximum:REAL; Default:STRING): REAL;
  VAR
    Answer:STRING;            { User response to screen prompt }
    Answer_as_Real:REAL;      { Conversion of STRING response to REAL number }
    Converted_OK:INTEGER;     { Confirmation of conversion to REAL number }
  BEGIN
  REPEAT
```

```
    REPEAT
      WRITE(Question);
      READLN(Answer);
      IF ((Answer='') AND (Default<>'')) THEN Answer:=Default;
      VAL(Answer,Answer_as_Real,Converted_OK);
    UNTIL (Converted_OK=0);
  UNTIL (Answer_as_real >= Minimum)
    AND (Answer_as_real <= Maximum);
  Accept_Real:=Answer_as_Real;
  END;


{ Comment: The function Ask_User prompts the user for a YES/NO response
  to a locally-specified question. The user response is converted to
  upper case, and must be YES, NO, Y or N. The question is repeated
  until one of these values is given. The function returns a Boolean
  variable; YES/Y giving a "TRUE" value, NO/N giving a "FALSE" value. }

FUNCTION Ask_User(Question:STRING):BOOLEAN;
  VAR
    Answer: STRING[3];
    Counter: INTEGER;
  BEGIN
    REPEAT
      WRITE(Question);
      READLN(Answer);
      FOR Counter:=1 TO 3 DO Answer[Counter]:=UPCASE(Answer[Counter]);
    UNTIL (Answer='Y') OR
          (Answer='YES') OR
          (Answer='N') OR
          (Answer='NO');
    Ask_User := ((Answer='Y') OR (Answer='YES'));
  END;


{ Comment: The function Ask_Quit uses a locally-specified question
  to prompt the user for the character value Q or q. The function returns
  a Boolean value of "FALSE" unless Q or q is entered as a response. }

FUNCTION Ask_Quit(Question:STRING):BOOLEAN;
  VAR
    Answer: CHAR;
  BEGIN
    WRITE(Question);
    Answer := READKEY;
    Ask_Quit := ((Answer='Q') OR (Answer='q'));
  END;


{ Comment: Discontinue is a complex function, calling antoher function
  from within itself. In seeking a character response of Q or q,
  Discontinue utilises the function Ask_Quit. The function returns
  the Boolean value "TRUE" if the user response Q is offered.
  Discontinue is used to interrupt the model at an intermediate stage
  of execution. }

FUNCTION Discontinue(Instructions:STRING):BOOLEAN;
  BEGIN
  Message_X1 := 10;
  Message_Y1 := 48;
  Message_X2 := 74;
  Message_Y2 := 50;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  Discontinue := Ask_Quit(Instructions);
  END;
```

```
{ Comment: Exit_Options is unusually large for a function, and might
  arguably have been coded as a procedure. Following conclusion or
  interruption (via function "Discontinue"), Exit_Options presents a
  menu summarising the options available to the user. These character
  responses then operate as switches to control program execution. }

FUNCTION Exit_Options:CHAR;
  VAR
    Response: CHAR;
    Counter: INTEGER;
  BEGIN
  Counter:=0;
  TEXTMODE(co80);
  Message_X1 := 5;
  Message_Y1 := 3;
  Message_X2 := 75;
  Message_Y2 := 4;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  TEXTBACKGROUND(Black);
  TEXTCOLOR(WHITE);
  CLRSCR;
  WRITE
  ('_____EXIT OR CONTINUE_____');
  Message_X1 := 5;
  Message_Y1 := 5;
  Message_X2 := 75;
  Message_Y2 := 20;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  IF Colour_Display=TRUE THEN
    TEXTBACKGROUND(Red) ELSE
    TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITELN('Please select an option from those below');
  WRITELN;
  WRITELN('Enter C to continue with current simulation');
  WRITELN('      V to view the data output file');
  WRITELN('      R to repeat the simulation using the current parameters');
  WRITELN('      S to restart the simulation using new parameters');
  WRITELN('      X to exit');
  REPEAT
    WRITELN;
    IF (Counter >= 1)
    THEN WRITE('Insufficient user IQ. Please re-enter your selection: ')
    ELSE WRITE('Please enter your selection: ');
    Response:=READKEY;
    Response:=UPCASE(Response);
    INC(Counter);
  UNTIL (Response='C')
     OR (Response='V')
     OR (Response='R')
     OR (Response='S')
     OR (Response='X');
  Exit_Options := Response;
  END;


{ Comment: The following procedures, Ask_***, present a simple user-
  interface screen requesting input from the user. The Ask_*** procedures
  in general untilise the above functions to accept a variable value from
  the user for subsequent use within the simulation. The procedures are
  intended to exhibit a common general structure and operation. }

PROCEDURE Ask_Display_Type;
  BEGIN
  CLRSCR;
  Message_X1 := 5;
  Message_Y1 := 3;
  Message_X2 := 75;
```

```
  Message_Y2 := 4;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITE
  ('_____DISPLAY_____');
  Message_X1 := 5;
  Message_Y1 := 5;
  Message_X2 := 75;
  Message_Y2 := 20;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITELN;
  WRITELN
  ('You may wish to display graphical output from the model in black and');
  WRITELN
  ('white.  This would be appropriate if you have a monochrome monitor, or');
  WRITELN
  ('should you wish to take screen-dumps of the output to a printer.');
  WRITELN
  ('Do you wish to display the output in colour?');
  Colour_Display:=Ask_User
    ('Enter Y for colour display, or N for monochrome ');
  CLRSCR;
  END;


PROCEDURE Ask_Field_Dimensions;
  BEGIN
  CLRSCR;
  Message_X1 := 5;
  Message_Y1 := 3;
  Message_X2 := 75;
  Message_Y2 := 4;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITE
  ('_____FIELD SIZE_____');
  Message_X1 := 5;
  Message_Y1 := 5;
  Message_X2 := 75;
  Message_Y2 := 20;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  IF Colour_Display=TRUE THEN
    TEXTBACKGROUND(Green) ELSE
    TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITELN;
  WRITELN
  ('In this window you are asked to specify the size of the field (area)');
  WRITELN
  ('in which the disease will develop.  A small field will produce a short');
  WRITELN
  ('epidemic.  The field size is determined by entering the number of rows');
  WRITELN
  ('and columns (Y and X values respectively).  If you type 20 in response');
  WRITELN
  ('to both the Y= and X= prompts you will produce a square field of 400');
  WRITELN
  ('plants.  The maximum size allowed is Y = 40 and X = 50.');
  WRITELN;
  WRITELN
  ('Enter you choices below (press <ENTER> after each one):');
```

```
  Max_Y:=ACCEPT_INTEGER('Y =  ',1,40,'');
  WRITELN;
  Max_X:=ACCEPT_INTEGER('X =  ',1,50,'');
  END;


PROCEDURE Ask_Infection_Origin;
  VAR
    Specified_Focus: BOOLEAN;
  BEGIN
  CLRSCR;
  Message_X1 := 5;
  Message_Y1 := 3;
  Message_X2 := 75;
  Message_Y2 := 4;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITE
  ('_____PRIMARY INFECTION FOCUS_____');
  Message_X1 := 5;
  Message_Y1 := 5;
  Message_X2 := 75;
  Message_Y2 := 20;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  IF Colour_Display=TRUE THEN
    TEXTBACKGROUND(Red) ELSE
    TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITELN;
  WRITELN
  ('The epidemic will start from one point in the field; the primary');
  WRITELN
  ('infection focus.  The rate of the epidemic might be determined by the');
  WRITELN
  ('position of this focus in the field.  You can either select the point');
  WRITELN
  ('of the primary focus yourself, or allow the program to choose a point');
  WRITELN
  ('at random.  If you want to make the selection you will be asked');
  WRITELN
  ('for the co-ordinates of the point in the format Y =(row), X =(column).');
  WRITELN;
  Specified_Focus:=Ask_User
    ('Enter Y to specify co-ordinates, or N to allow a random choice: ');
  IF Specified_Focus THEN
    BEGIN
    WRITELN;
    WRITELN
    ('Enter the Y and X coordinates of the primary infection focus');
    WRITE
    ('NOTE: The centre of the field is at (');
    WRITE(Max_Y DIV 2);
    WRITE(',');
    WRITE(Max_X DIV 2);
    WRITELN(')');
    WRITE
    ('      The permitted range is from (1,1) to (');
    WRITE(Max_Y);
    WRITE(',');
    WRITE(Max_X);
    WRITELN(')');
    WRITE('Enter the Y coordinate (1 to ');
    WRITE(Max_Y);
    WRITELN(') and press <ENTER>:');
    Y0:=ACCEPT_INTEGER('Y = ',1,Max_Y,'');
    WRITELN;
```

```
    WRITE('Enter the X coordinate (1 to ');
    WRITE(Max_X);
    WRITELN(') and press <ENTER>:');
    X0:=ACCEPT_INTEGER('X = ',1,Max_X,'');
    END
  ELSE
    BEGIN
    Y0:=RANDOM(Max_Y)+1;
    X0:=RANDOM(Max_X)+1;
    END;
  END;


PROCEDURE Ask_Mode_of_Pathogen_Spread;
  VAR
    Answer: STRING[2];
    Counter: INTEGER;
  BEGIN
  CLRSCR;
  Message_X1 := 5;
  Message_Y1 := 5;
  Message_X2 := 75;
  Message_Y2 := 4;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITE
  ('_____MODE OF PATHOGEN DISPERSAL_____');
  Message_X1 := 5;
  Message_Y1 := 5;
  Message_X2 := 75;
  Message_Y2 := 20;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  IF Colour_Display=TRUE THEN
    TEXTBACKGROUND(Red) ELSE
    TEXTCOLOR(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITELN;
  WRITELN
  ('The mode of dispersal of a pathogen has a profound effect on the form');
  WRITELN
  ('of epidemic which it can cause.  The models presented here represent');
  WRITELN
  ('the common simplifications which are used in many general text books.');
  WRITELN
  ('Thus, the SOILBORNE pathogen spreads mainly between neighbours');
  WRITELN
  ('and from only one focus which expands over a number of cycles, while');
  WRITELN
  ('the AIRBORNE and SPLASH dispersed pathogens spread both from neighbour');
  WRITELN
  ('to neighbour and from the development of new foci.');
  WRITELN;
  WRITELN
  ('Beware of these generalisations when considering real data!');
  WRITELN;
  REPEAT
    WRITELN('Enter S for SOILBORNE,');
    WRITELN;
    WRITELN('      A for Polycyclic AIRBORNE,');
    WRITELN;
    WRITE  ('      W for Polcyclic SPLASH dispersed.  ');
    READLN(Answer);
    FOR Counter:=1 TO 2 DO Answer[Counter]:=UPCASE(Answer[Counter]);
  UNTIL
    (Answer='S') OR (Answer='A') OR (Answer='W');
  IF (Answer='A') THEN Mode_of_Pathogen_Spread:='AIRBORNE'
```

```
  ELSE
  IF (Answer='W') THEN Mode_of_Pathogen_Spread:='SPLASH'
  ELSE
  Mode_of_Pathogen_Spread:='SOILBORNE';
  END;

PROCEDURE Ask_Temperature_Conditions;
  BEGIN
  CLRSCR;
  Message_X1 := 5;
  Message_Y1 := 3;
  Message_X2 := 75;
  Message_Y2 := 4;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITE
  ('_____TEMPERATURE CONDITIONS_____');
  Message_X1 := 5;
  Message_Y1 := 5;
  Message_X2 := 75;
  Message_Y2 := 20;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  IF Colour_Display=TRUE THEN
    TEXTBACKGROUND(Blue) ELSE
    TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITELN;
  WRITELN
  ('Temperature is the dominant environmental variable in many plant');
  WRITELN
  ('disease epidemics.  In this simulation program the diseases are fungal');
  WRITELN
  ('pathogens which have temperature optima of around 20 C.');
  WRITELN
  ('If the temperature is above or below its optimum the success rate');
  WRITELN
  ('of attempted infections decreases, so some plants which might be');
  WRITELN
  ('infected by a given batch of spores escape infection, and the overall');
  WRITELN
  ('rate of the epidemic is reduced.');
  WRITELN;
  WRITELN
  ('The temperature does not stay constant over the period of the epidemic');
  WRITELN
  ('and you will be asked to specify the MEAN temperature and the standard');
  WRITELN
  ('deviation about this mean value.  The greater the standard deviation');
  WRITELN
  ('the more likely it is that unfavourable conditions for the pathogen');
  WRITELN
  ('will occur.');
  WRITELN;
  Mean_Temperature:=
    ACCEPT_REAL
    ('Enter the MEAN temperature, between 0 and 35 inclusive ',0,35,'');
  WRITELN;
  Temperature_sd:=
    ACCEPT_REAL
    ('Enter the standard deviation of temperature across the season ',0,10,'');
  END;
```

```
PROCEDURE Ask_WindDirection;
  VAR
    GO : CHAR;
    counter: INTEGER;
    Answer: STRING[2];
    Variable_WindDirection: BOOLEAN;
  BEGIN
  CLRSCR;
  Message_X1 := 5;
  Message_Y1 := 3;
  Message_X2 := 75;
  Message_Y2 := 4;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITE
  ('_____WIND DIRECTION AND SPEED_____');
  Message_X1 := 5;
  Message_Y1 := 5;
  Message_X2 := 75;
  Message_Y2 := 20;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  IF Colour_Display=TRUE THEN
    TEXTBACKGROUND(Blue) ELSE
    TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITELN;
  WRITELN
  ('The dispersal of air borne diseases can be influenced by the direction');
  WRITELN
  ('and speed of the wind.');
  WRITELN
  ('Watch the pattern of dispersal carefully; the wind may not influence');
  WRITELN
  ('the dispersal of the disease in quite the way you expect.');
  WRITELN;
  WRITELN
  ('You are asked for the prevailing wind direction and whether you want');
  WRITELN
  ('the wind to blow from a constant direction or to vary.  If you specify');
  WRITELN
  ('variation in wind direction you will be asked to define the range of');
  WRITELN
  ('possible wind directions as an arc about the prevailing direction.');
  WRITELN;
  WRITELN
  ('The wind variables are entered in the following window.');
  WRITELN;
  WRITELN;
  WRITELN
  ('                    press <ENTER> to continue');
  GO:=readkey;
  CLRSCR;
  WRITELN;
  WRITELN
  ('Please enter wind direction in abbreviated form, eg. SW for South West.');
  WRITELN
  ('Enter:  N, for Northerly wind');
  WRITELN
  ('        NE, for North-Easterly wind');
  WRITELN
  ('        E, for Easterly wind');
  WRITELN
  ('        SE, for South-Easterly wind');
  WRITELN
  ('        S, for Southerly wind');
  WRITELN
```

```
  ('        SW, for South-Westerly wind');
  WRITELN
  ('        W, for Westerly wind');
  WRITE
  ('        NW, for North-Westerly wind');
  REPEAT
    BEGIN
    WRITE('Mean Wind direction = ');
    READLN(Answer);
    FOR counter:=1 TO 2 DO Answer[counter]:=UPCASE(Answer[counter]);
    END;
  UNTIL
    (Answer='N') OR (Answer='NE') OR
    (Answer='E') OR (Answer='SE') OR
    (Answer='S') OR (Answer='SW') OR
    (Answer='W') OR (Answer='NW');
  Mean_WindDirection:=Answer;
  WRITELN;
  WRITELN
  ('Will there be a variation in wind direction? ');
  WRITELN;
  Variable_WindDirection:=
    Ask_User('Enter Y for variation, or N for constant wind direction. ');
  IF Variable_WindDirection THEN
    BEGIN
    WRITELN;
    WRITELN
    ('Please specify the range of possible wind directions in the form of');
    WRITELN
    ('an arc about the mean wind direction.');
    REPEAT
      WRITELN;
      Wind_Arc:=ACCEPT_INTEGER
      ('Values of 0, 90, 180, 270 and 360 degrees will be accepted: ',0,360,'');
    UNTIL
      (Wind_Arc=0) OR (Wind_Arc=90) OR (Wind_Arc=180) OR
      (Wind_Arc=270) OR (Wind_Arc=360);
    END
  ELSE Wind_Arc:=0;
  END;


PROCEDURE Ask_WindSpeed;
  VAR
    Go: CHAR;
    Specified_Threshold: BOOLEAN;
  BEGIN
  WindSpeed_Threshold:=0;
  IF Mode_of_Pathogen_Spread='AIRBORNE' THEN
  BEGIN
  CLRSCR;
  WRITELN;
  WRITELN
  ('In some cases spores of a pathogen may not be released from infected');
  WRITELN
  ('plants unless there is sufficient air movement.  Wind speed also has');
  WRITELN
  ('an influence on the distance which spores are dispersed and the');
  WRITELN
  ('efficiency with which they impact on the on the leaf surface.');
  WRITELN;
  WRITELN
  ('You will be asked whether there is a minimum wind speed below which');
  WRITELN
  ('spores cannot be released.  If you select this option, variation in');
  WRITELN
  ('wind speed will affect the number of secondary infections formed at');
  WRITELN
  ('each cycle of the epidemic.  If you decide that the pathogen can');
```

```
  WRITELN
  ('release its spores independent of air movement, the speed of the wind');
  WRITELN
  ('will still influence the distance from an infected plant over which');
  WRITELN
  ('spores can be dispersed.');
  WRITELN;
  WRITELN
  ('                    press <ENTER> to continue');
  Go:= READKEY;
  CLRSCR;
  WRITELN;
  WRITELN
  ('Is there a threshold wind speed, below which spores will not be');
  WRITELN
  ('released? ');
  WRITELN;
  Specified_Threshold:=Ask_User
    ('Enter Y to specify a threshold speed, or N for none: ');
  IF Specified_Threshold THEN
    BEGIN
    WRITELN;
    WRITELN
    ('Threshold values between 0 and 10 metres per second will be accepted.');
    WindSpeed_Threshold:=ACCEPT_REAL
      ('What is the threshold wind (in m/s) for spore release? ',0,10,'');
    END
  ELSE
    WindSpeed_Threshold:=0;
  END;
  CLRSCR;
  WRITELN;
  WRITELN
  ('Now enter the MEAN and standard deviation of the wind speed during the');
  WRITELN
  ('epidemic.');
  WRITELN
  ('What is the MEAN wind speed (m/s) over the season?');
  Mean_WindSpeed:=ACCEPT_REAL
    ('Real values between 0 and 10 will be accepted: ',0,10,'');
  WRITELN;
  WRITELN
    ('What is the standard deviation in wind speed?');
  WindSpeed_sd:=ACCEPT_REAL
    ('Real values between 0 and 5 will be accepted: ',0,5,'');
  END;


PROCEDURE Ask_Rainfall;
  VAR
    Answer: STRING[2];
    counter: INTEGER;
  BEGIN
  CLRSCR;
  Message_X1 := 5;
  Message_Y1 := 3;
  Message_X2 := 75;
  Message_Y2 := 4;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITE
  ('_____RAINFALL_____');
  Message_X1 := 5;
  Message_Y1 := 5;
  Message_X2 := 75;
  Message_Y2 := 20;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
```

```
  IF Colour_Display=TRUE THEN
    TEXTBACKGROUND(Blue) ELSE
    TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITELN;
  WRITELN
  ('For splash-dispersed pathogens the amount of rainfall is obviously');
  WRITELN
  ('important in determining the rate of epidemics.  You can introduce a');
  WRITELN
  ('crude simulation of this effect by selecting one of three broad types');
  WRITELN
  ('of rainfall pattern.');
  WRITELN;
  WRITELN;
  REPEAT
    WRITELN('Enter D for a DRY season,');
    WRITELN;
    WRITELN('      T for a TYPICALly wet season,');
    WRITELN;
    WRITE  ('      W for a very WET season.          ');
    READLN(Answer);
    FOR counter:=1 TO 2 DO Answer[counter]:=UPCASE(Answer[counter]);
  UNTIL
    (Answer='D') OR (Answer='T') OR (Answer='W');
  IF (Answer='W') THEN Rainfall:='HEAVY';
  IF (Answer='T') THEN Rainfall:='TYPICAL';
  IF (Answer='D') THEN Rainfall:='LIGHT';
  END;


PROCEDURE Ask_Host_Susceptibility;
  BEGIN
  CLRSCR;
  Message_X1 := 5;
  Message_Y1 := 3;
  Message_X2 := 75;
  Message_Y2 := 4;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITE
  ('_____EFFECTS OF UNDEFINED VARIABLES_____');
  Message_X1 := 5;
  Message_Y1 := 5;
  Message_X2 := 75;
  Message_Y2 := 20;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  IF Colour_Display=TRUE THEN
    TEXTBACKGROUND(Cyan) ELSE
    TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITELN;
  WRITELN
  ('It has been observed frequently that even when a genetically uniform');
  WRITELN
  ('pathogen infects a genetically uniform host variety under uniform and');
  WRITELN
  ('suitable conditions some of the spores of the pathogen do not show');
  WRITELN
  ('typical development patterns.  There are probably a large number of');
  WRITELN
  ('unexplained interactions which cause these "random" effects.');
  WRITELN
  ('You can introduce a degree of uncertainty into the epidemic to mimic');
  WRITELN
```

```
  ('these random effects.  You will be asked for a probability factor');
  WRITELN
  ('(50-100%) which will determine the chances of a spore infecting a plant');
  WRITELN
  ('after the effects of all of the defined environmental variables have');
  WRITELN
  ('taken into account.  If you do not want to introduce this effect you');
  WRITELN
  ('should enter a value of 100.');
  WRITELN;
  Chance_susceptible:=
    ACCEPT_INTEGER
    ('Infection success rate with "random" factors: ',50,100,'100');
  WRITELN;
  CLRSCR;
  END;


PROCEDURE Ask_Differentiate_Infectious;
  BEGIN
  CLRSCR;
  Message_X1 := 5;
  Message_Y1 := 3;
  Message_X2 := 75;
  Message_Y2 := 4;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITE
  ('_____VISUAL OUTPUT_____');
  Message_X1 := 5;
  Message_Y1 := 5;
  Message_X2 := 75;
  Message_Y2 := 20;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  IF Colour_Display=TRUE THEN
    TEXTBACKGROUND(BROWN) ELSE
    TEXTBACKGROUND(BLACK);
  TEXTCOLOR(white);
  CLRSCR;
  WRITELN;
  WRITELN
  ('Do you wish to be able to visually differentiate between INFECTED and');
  WRITELN
  ('INFECTIOUS plants in the field?');
  See_infectious:=Ask_User
    ('Enter Y for indentifiable INFECTIOUS, of N for indistingishable ');
  CLRSCR;
  END;


{ Comment: The procedure User_Specified_Parameters is responsible for
  the specifting the execution order of the Ask_*** procedures. This
  simplifies the task of amending and adding to the procedures called.
  This procedure is called from withing the procedure
  Accept_Specified_Parameters. }

PROCEDURE User_Specified_Parameters;
  BEGIN
  CLRSCR;
  Ask_Display_Type;
  Ask_Field_Dimensions;
    Max_Number_of_Plants_in_Field := Max_Y * Max_X;
  Ask_Infection_Origin;
  Ask_Temperature_Conditions;
  Ask_Mode_of_Pathogen_Spread;
  IF (Mode_of_Pathogen_Spread='AIRBORNE') THEN
    BEGIN
```

```
    Ask_WindDirection;
    Ask_WindSpeed;
    END;
  IF (Mode_of_Pathogen_Spread='SPLASH') THEN
    BEGIN
    Ask_WindDirection;
    Ask_Rainfall;
    END;
  Ask_Host_Susceptibility;
  Ask_Differentiate_INFECTIOUS;
  END;


{ Comment: After executing the procedure User_Specified_Parameters,
  the procedure Accept_Specified_Parameters displays a summary of the
  user-specified variables obtained through the Ask_*** series of
  procedures. The summary then asks the user to accept the displayed
  variables (at this stage the function Ask_User might have been used)
  by entering a sting value of YES/Y, or to reject the variables (NO/N)
  and repeat the input procedures. }

PROCEDURE Accept_Specified_Parameters;
  VAR
    Summary_Approved: STRING[3];
    counter: INTEGER;
  BEGIN
  CLRSCR;
  REPEAT
    BEGIN
    User_Specified_Parameters;
    Message_X1 := 5;
    Message_Y1 := 3;
    Message_X2 := 75;
    Message_Y2 := 4;
    WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
    TEXTBACKGROUND(Black);
    TEXTCOLOR(White);
    CLRSCR;
    WRITE
    ('_____SUMMARY OF SETTINGS_____');
    Message_X1 := 5;
    Message_Y1 := 5;
    Message_X2 := 75;
    Message_Y2 := 20;
    WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
    IF Colour_Display=TRUE THEN
      TEXTBACKGROUND(Blue) ELSE
      TEXTBACKGROUND(Black);
    TEXTCOLOR(White);
    CLRSCR;
    WRITELN
    ('SUMMARY OF USER-DEFINED VARIABLES');
    WRITELN;
    WRITELN
    ('Field dimensions of ',Max_Y,' by ',Max_X,' plants, giving a total of
',Max_Number_of_Plants_in_Field,' plants');
    WRITELN
    ('Primary infection focus at plant ',Y0,', ',X0);
    WRITELN
    ('Mean Temperature = ',Mean_Temperature:4:1,' degrees centigrade; sd =
',Temperature_sd:3:1);
    WRITELN
    ('Mode of pathogen dispersal = ',Mode_of_Pathogen_Spread);
    IF (mode_of_pathogen_spread='AIRBORNE') THEN
      BEGIN
      WRITELN('Prevailing wind is from the ',Mean_WindDirection);
      WRITELN('Mean Wind Speed = ',Mean_WindSpeed:4:1,' m/s');
      END;
    IF (mode_of_pathogen_spread='SPLASH') THEN
```

```
      BEGIN
      WRITELN('Prevailing wind is from the ',Mean_WindDirection);
      WRITELN('Rainfall is ',Rainfall);
      END;
    WRITELN
    ('There is a ',100-Chance_susceptible,'% chance of infection failing at random');
    WRITELN;
    WRITELN
    ('Enter Y to proceed with the simulation or N to return to the');
    WRITELN
    ('parameter input procedure.');
    WRITELN;
    REPEAT
      BEGIN
      WRITE('PROCEED WITH SIMULATION? ');
      READLN(Summary_approved);
      FOR counter:=1 TO 3 DO
        Summary_approved[counter]:=UPCASE(Summary_approved[counter]);
      END;
    UNTIL (Summary_approved='Y') OR
          (Summary_approved='YES') OR
          (Summary_approved='N') OR
          (Summary_approved='NO');
    END;
  UNTIL (Summary_approved='Y') OR (Summary_approved='YES');
  END;


{ Comment: The procedure Ask_Output_Filename is executed separately from
  the rest of the Ask_*** procedures. This allows for the specification
  of a different output file name for each run of the simulation, even
  in the event of the user opting to repeat the simulation using the
  user-specified varibales of the preceding run. }

PROCEDURE Ask_Output_Filename;
  BEGIN
  CLRSCR;
  Message_X1 := 5;
  Message_Y1 := 3;
  Message_X2 := 75;
  Message_Y2 := 4;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITE
  ('_____OUTPUT FILE_____');
  Message_X1 := 5;
  Message_Y1 := 5;
  Message_X2 := 75;
  Message_Y2 := 20;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  IF Colour_Display=TRUE THEN
    TEXTBACKGROUND(Brown) ELSE
    TEXTBACKGROUND(Black);
  TEXTCOLOR(White);
  CLRSCR;
  WRITELN;
  WRITELN
  ('An output file, displaying numerical values for the epidemic');
  WRITELN
  ('parameters, will be produced. This file will be written to the');
  WRITELN
  ('directory in which the executable file resides. The output file is');
  WRITELN
  ('bound by DOS nomenclature rules: 1-8 characters, no spaces,');
  WRITELN
  ('punctuation or special characters. The file will be given the file');
  WRITELN
```

```
  ('extension, ".DAT". Please enter the filename.');
  WRITELN;
  REPEAT
    BEGIN
    WRITE('Name for data output file: ');
    READLN(Data_OutputFile_Name);
    END;
  UNTIL (Data_OutputFile_Name <> '');
  CLRSCR;
  END;


{ Comment: This procedure utilises a feature called "String Concatenation"
  to join the variable Data_OutputFile_Name (specified in the procedure,
  Ask_Output_Filename) to a file extension suffix, and thus specify the
  complete DOS filename for data output. After performing the necessary
  PASCAL operations of ASSIGNing and Opening (REWRITE) a new file, the
  procedure establishes some column headers and other bits'n'bobs. }

PROCEDURE Open_Data_Output;
  BEGIN
  ASSIGN(Data_Output,Data_OutputFile_Name+'.DAT');
  REWRITE(Data_Output);
  WRITELN
  (Data_Output,'File: '+Data_OutputFile_Name+'.DAT.');
  WRITELN
  (Data_Output,'Program '+Program_Name+'.PAS results file');
  WRITELN(Data_Output);
  WRITELN
  (Data_Output,'Number of plants in field = ',Max_Number_of_Plants_in_Field:4);
  WRITELN
  (Data_Output,'Primary infection focus at Y=',Y0:3,', X=',X0:3);
  WRITE
  (Data_Output,'Mean_Temperature = ',Mean_Temperature:4:1,chr(167)+'C; ');
  WRITELN
  (Data_Output,'s.d. = ',Temperature_sd:5:2);
  WRITELN
  (Data_Output,'Mode of pathogen dispersal = ',Mode_of_Pathogen_Spread);
  IF (Mode_of_Pathogen_Spread='AIRBORNE') THEN
    BEGIN
    WRITE
    (Data_Output,'Prevailing wind direction = ',Mean_WindDirection,'; ');
    WRITELN
    (Data_Output,'Arc = ',Wind_Arc,chr(167));
    WRITE
    (Data_Output,'Mean wind speed = ',Mean_WindSpeed:5:2,'m/s; ');
    WRITELN
    (Data_Output,'s.d. = ',WindSpeed_sd:5:2);
    END;
  IF (Mode_of_Pathogen_Spread='SPLASH') THEN
    BEGIN
    WRITE
    (Data_Output,'Prevailing wind direction = ',Mean_WindDirection,'; ');
    WRITELN
    (Data_Output,'Arc = ',Wind_Arc,chr(167));
    WRITELN
    (Data_Output,'Rainfall = ',Rainfall);
    END;
  WRITELN
  (Data_Output,'Chance of infection failing = ',(100-Chance_susceptible):2,'%');
  WRITELN(Data_Output);
  WRITELN(Data_Output,
  'Cycle/ Plants:                          Temp. Wind          P(a)');
  WRITELN(Data_Output,
  ' Phase Healthy Infected Infectious Dead          Dir. Speed');
  WRITELN(Data_Output);
  END;
```

```
PROCEDURE Close_Data_Output;
  BEGIN
  APPEND(Data_output);
  WRITELN(Data_output);
  WRITELN(Data_output,'END-OF-FILE');
  CLOSE(Data_output);
  END;


PROCEDURE Open_Dbug_Output;
  BEGIN
  ASSIGN(DBug_output,Program_name+'.BUG');
  REWRITE(DBug_output);
  WRITELN
  (DBug_output,'File: '+Program_name+'.BUG.');
  WRITELN
  (DBug_output,'Program '+Program_name+' execution progression record');
  WRITELN
  (DBug_output,' ');
  END;


PROCEDURE Close_DBug_Output;
  BEGIN
  CLOSE(DBug_output);
  END;


{ Comment: Operating on a user-specified mean temperature and its
  standard deviation, a realised temperature for a given pathogen growth
  cycle is obtained by randomly selection from the normal distribution
  defined by the mean and standard deviation values.  System-supplied
  uniform random values are transformed to a normal distribution by one
  of three methods: Box-Muller (1958, Ann.Math.Stat.,29), Marsaglia-
  Bray polar transformation (1964, SIAM Rev.,6) or Central Limit
  Theorem. Marsaglia-Bray is currently the prefered method. }

PROCEDURE Temperature_Effect;
  VAR
    U1,U2,U3,V1,V2: REAL;
    Counter,n: INTEGER;
  BEGIN
    CASE (RANDOM(100)+1) OF
      1..86   : BEGIN
                U1:=(RANDOM(1000)+1)/1000;
                U2:=(RANDOM(1000)+1)/1000;
                U3:=(RANDOM(1000)+1)/1000;
                Temperature_Deviation:=Temperature_sd*2*(U1+U2+U3-1.5);
                END;
      87..97  : BEGIN
                U1:=(RANDOM(1000)+1)/1000;
                U2:=(RANDOM(1000)+1)/1000;
                Temperature_Deviation:=Temperature_sd*1.5*(U1+U2-1);
                END;
      98..100 : BEGIN
                REPEAT
                  V1:=(RANDOM(1001)/500)-1;
                  V2:=(RANDOM(1001)/500)-1;
                UNTIL ( SQR(V1)+SQR(V2) < 1 );
                  Temperature_Deviation:=Temperature_sd*V1
                                         *SQRT(-2*ln(SQR(V1)+SQR(V2))
                                         /(SQR(V1)+SQR(V2)));
                END;
    END; {CASE}
  Realised_Temperature:=(Mean_Temperature+Temperature_Deviation);

{ Comment: Pathogen activity is equated to temperature through a simple
  polynomial function. Neil devised this one. He's a clever chap! We can
  expect the relationship between temperature and pathogen activity to
```

```
  be pathogen-specific, and it might be an interesting exercise to see
  how different functions affect the model output. }

  Probability_of_pathogen_growth:=
    3.5+(10*Realised_Temperature)-(0.26*SQR(Realised_Temperature));

{ Comment: The variable Probability_of_Pathogen_Growth is used directly
  in the Primary_Infection procedure, governing the infection of plants
  adjacent (or in close proximety) to and infectious plant. The P_o_P_G
  value is translated to a "Chance_of_Secondary_Infection" for use in the
  Secondary_Infection procedure, governing remote infections. }

  IF Probability_of_Pathogen_Growth < 0
    THEN Probability_of_Pathogen_Growth:=0;
  IF Probability_of_Pathogen_Growth >100
    THEN Probability_of_Pathogen_Growth:=100;

  IF Probability_of_Pathogen_Growth < 33
    THEN Secondary_Infection_Potential:=1
  ELSE IF Probability_of_Pathogen_Growth > 66
    THEN Secondary_Infection_Potential:=4
  ELSE Secondary_Infection_Potential:=2;
{ Chance_of_Secondary_Infection:=
    ROUND((Secondary_Infection_Potential*Probability_of_Pathogen_Growth)/4); }
  Chance_of_Secondary_Infection:=
    (Secondary_Infection_Potential*Probability_of_Pathogen_Growth)/4;
  END;


{ Comment: The user-specified string variable (restricted to the 8 major
  compass points: N, NE, E, SE... NW) accepted by the Ask_WindDirection
  procedure, is taken as the prevailing wind direction (here referred to
  as Mean_WindDirection). This forms the mean about which the wind direction
  will vary. The range of variation is given by the Wind_Arc variable.
  We can expect the Realised_WindDirection for each pathogen cycle to be
  normally distributed about the prevailing direction. A
  Realised_WindDirection value is generated by a call to the compiler
  RANDOM function. Due to the digitised (non-continuous) representation of
  wind direction (compass points as opposed to 0-360 degrees), the
  distribution is only an approximation of the normal.
  The procedure WindDirection_Effect is physically longer than might be
  desirable. String variables cannot be used is a "CASE OF" structure, and
  so the procedure, which intuitively should be coded as a 2-level nested
  case, is comprised of a series of IF-THEN-ELSE wind direction arguments,
  within which integer Wind_Arc values are arranged in a CASE structure.
  The RInt99 value ranges defining the distribution are by necessity
  hard-coded into the program, which is far from desirable. }

PROCEDURE WindDirection_Effect;
  VAR
    RInt99: INTEGER;
  BEGIN
  RInt99:=RANDOM(100);
    IF Mean_WindDirection = 'N' THEN
    BEGIN
      CASE Wind_Arc OF
        0:   Realised_WindDirection:='N';
        90:  CASE RInt99 OF
               0..24:  Realised_WindDirection:='NW';
               25..74: Realised_WindDirection:='N';
               75..99: Realised_WindDirection:='NE';
             END; {CASE}
        180: CASE RInt99 OF
               0..9:   Realised_WindDirection:='W';
               10..29: Realised_WindDirection:='NW';
               30..69: Realised_WindDirection:='N';
               70..89: Realised_WindDirection:='NE';
               90..99: Realised_WindDirection:='E';
             END; {CASE}
```

```
      270: CASE RInt99 OF
             0..4:   Realised_WindDirection:='SW';
             5..13:  Realised_WindDirection:='W';
             14..32: Realised_WindDirection:='NW';
             33..65: Realised_WindDirection:='N';
             66..84: Realised_WindDirection:='NE';
             85..93: Realised_WindDirection:='E';
             94..99: Realised_WindDirection:='SE';
           END; {CASE}
      360: CASE RInt99 OF
             0..2:   Realised_WindDirection:='S';
             3..6:   Realised_WindDirection:='SW';
             7..14:  Realised_WindDirection:='W';
             15..30: Realised_WindDirection:='NW';
             31..61: Realised_WindDirection:='N';
             62..87: Realised_WindDirection:='NE';
             88..95: Realised_WindDirection:='E';
             96..99: Realised_WindDirection:='SE';
           END; {CASE}
    END; {CASE}
  END
ELSE
IF Mean_WindDirection = 'NE' THEN
  BEGIN
    CASE Wind_Arc OF
      0:   Realised_WindDirection:='NE';
      90:  CASE RInt99 OF
             0..24:  Realised_WindDirection:='N';
             25..74: Realised_WindDirection:='NE';
             75..99: Realised_WindDirection:='E';
           END; {CASE}
      180: CASE RInt99 OF
             0..9:   Realised_WindDirection:='NW';
             10..29: Realised_WindDirection:='N';
             30..69: Realised_WindDirection:='NE';
             70..89: Realised_WindDirection:='E';
             90..99: Realised_WindDirection:='SE';
           END; {CASE}
      270: CASE RInt99 OF
             0..4:   Realised_WindDirection:='W';
             5..13:  Realised_WindDirection:='NW';
             14..32: Realised_WindDirection:='N';
             33..65: Realised_WindDirection:='NE';
             66..84: Realised_WindDirection:='E';
             85..93: Realised_WindDirection:='SE';
             94..99: Realised_WindDirection:='S';
           END; {CASE}
      360: CASE RInt99 OF
             0..2:   Realised_WindDirection:='SW';
             3..6:   Realised_WindDirection:='W';
             7..14:  Realised_WindDirection:='NW';
             15..30: Realised_WindDirection:='N';
             31..61: Realised_WindDirection:='NE';
             62..87: Realised_WindDirection:='E';
             88..95: Realised_WindDirection:='SE';
             96..99: Realised_WindDirection:='S';
           END; {CASE}
    END; {CASE}
  END
ELSE
IF Mean_WindDirection = 'E' THEN
  BEGIN
    CASE Wind_Arc OF
      0:   Realised_WindDirection:='E';
      90:  CASE RInt99 OF
             0..24:  Realised_WindDirection:='NE';
             25..74: Realised_WindDirection:='E';
             75..99: Realised_WindDirection:='SE';
           END; {CASE}
```

```
      180: CASE RInt99 OF
             0..9:   Realised_WindDirection:='N';
             10..29: Realised_WindDirection:='NE';
             30..69: Realised_WindDirection:='E';
             70..89: Realised_WindDirection:='SE';
             90..99: Realised_WindDirection:='S';
           END; {CASE}
      270: CASE RInt99 OF
             0..4:   Realised_WindDirection:='NW';
             5..13:  Realised_WindDirection:='N';
             14..32: Realised_WindDirection:='NE';
             33..65: Realised_WindDirection:='E';
             66..84: Realised_WindDirection:='SE';
             85..93: Realised_WindDirection:='S';
             94..99: Realised_WindDirection:='SW';
           END; {CASE}
      360: CASE RInt99 OF
             0..2:   Realised_WindDirection:='W';
             3..6:   Realised_WindDirection:='NW';
             7..14:  Realised_WindDirection:='N';
             15..30: Realised_WindDirection:='NE';
             31..61: Realised_WindDirection:='E';
             62..87: Realised_WindDirection:='SE';
             88..95: Realised_WindDirection:='S';
             96..99: Realised_WindDirection:='SW';
           END; {CASE}
    END; {CASE}
  END
ELSE
IF Mean_WindDirection = 'SE' THEN
  BEGIN
    CASE Wind_Arc OF
      0:   Realised_WindDirection:='SE';
      90:  CASE RInt99 OF
             0..24:  Realised_WindDirection:='E';
             25..74: Realised_WindDirection:='SE';
             75..99: Realised_WindDirection:='S';
           END; {CASE}
      180: CASE RInt99 OF
             0..9:   Realised_WindDirection:='NE';
             10..29: Realised_WindDirection:='E';
             30..69: Realised_WindDirection:='SE';
             70..89: Realised_WindDirection:='S';
             90..99: Realised_WindDirection:='SW';
           END; {CASE}
      270: CASE RInt99 OF
             0..4:   Realised_WindDirection:='N';
             5..13:  Realised_WindDirection:='NE';
             14..32: Realised_WindDirection:='E';
             33..65: Realised_WindDirection:='SE';
             66..84: Realised_WindDirection:='S';
             85..93: Realised_WindDirection:='SW';
             94..99: Realised_WindDirection:='W';
           END; {CASE}
      360: CASE RInt99 OF
             0..2:   Realised_WindDirection:='NW';
             3..6:   Realised_WindDirection:='N';
             7..14:  Realised_WindDirection:='NE';
             15..30: Realised_WindDirection:='E';
             31..61: Realised_WindDirection:='SE';
             62..87: Realised_WindDirection:='S';
             88..95: Realised_WindDirection:='SW';
             96..99: Realised_WindDirection:='W';
           END; {CASE}
    END; {CASE}
  END
ELSE
IF Mean_WindDirection = 'S' THEN
  BEGIN
```

```
    CASE Wind_Arc OF
      0:   Realised_WindDirection:='S';
      90:  CASE RInt99 OF
             0..24:  Realised_WindDirection:='SE';
             25..74: Realised_WindDirection:='S';
             75..99: Realised_WindDirection:='SW';
           END; {CASE}
      180: CASE RInt99 OF
             0..9:   Realised_WindDirection:='E';
             10..29: Realised_WindDirection:='SE';
             30..69: Realised_WindDirection:='S';
             70..89: Realised_WindDirection:='SW';
             90..99: Realised_WindDirection:='W';
           END; {CASE}
      270: CASE RInt99 OF
             0..4:   Realised_WindDirection:='NE';
             5..13:  Realised_WindDirection:='E';
             14..32: Realised_WindDirection:='SE';
             33..65: Realised_WindDirection:='S';
             66..84: Realised_WindDirection:='SW';
             85..93: Realised_WindDirection:='W';
             94..99: Realised_WindDirection:='NW';
           END; {CASE}
      360: CASE RInt99 OF
             0..2:   Realised_WindDirection:='N';
             3..6:   Realised_WindDirection:='NE';
             7..14:  Realised_WindDirection:='E';
             15..30: Realised_WindDirection:='SE';
             31..61: Realised_WindDirection:='S';
             62..87: Realised_WindDirection:='SW';
             88..95: Realised_WindDirection:='W';
             96..99: Realised_WindDirection:='NW';
           END; {CASE}
    END; {CASE}
  END
ELSE
IF Mean_WindDirection = 'SW' THEN
  BEGIN
    CASE Wind_Arc OF
      0:   Realised_WindDirection:='SW';
      90:  CASE RInt99 OF
             0..24:  Realised_WindDirection:='S';
             25..74: Realised_WindDirection:='SW';
             75..99: Realised_WindDirection:='W';
           END; {CASE}
      180: CASE RInt99 OF
             0..9:   Realised_WindDirection:='SE';
             10..29: Realised_WindDirection:='S';
             30..69: Realised_WindDirection:='SW';
             70..89: Realised_WindDirection:='W';
             90..99: Realised_WindDirection:='NW';
           END; {CASE}
      270: CASE RInt99 OF
             0..4:   Realised_WindDirection:='E';
             5..13:  Realised_WindDirection:='SE';
             14..32: Realised_WindDirection:='S';
             33..65: Realised_WindDirection:='SW';
             66..84: Realised_WindDirection:='W';
             85..93: Realised_WindDirection:='NW';
             94..99: Realised_WindDirection:='N';
           END; {CASE}
      360: CASE RInt99 OF
             0..2:   Realised_WindDirection:='NE';
             3..6:   Realised_WindDirection:='E';
             7..14:  Realised_WindDirection:='SE';
             15..30: Realised_WindDirection:='S';
             31..61: Realised_WindDirection:='SW';
             62..87: Realised_WindDirection:='W';
             88..95: Realised_WindDirection:='NW';
```

```
            96..99: Realised_WindDirection:='N';
          END; {CASE}
    END; {CASE}
  END
ELSE
IF Mean_WindDirection = 'W' THEN
  BEGIN
    CASE Wind_Arc OF
      0:   Realised_WindDirection:='W';
      90:  CASE RInt99 OF
             0..24:  Realised_WindDirection:='SW';
             25..74: Realised_WindDirection:='W';
             75..99: Realised_WindDirection:='NW';
           END; {CASE}
      180: CASE RInt99 OF
             0..9:   Realised_WindDirection:='S';
             10..29: Realised_WindDirection:='SW';
             30..69: Realised_WindDirection:='W';
             70..89: Realised_WindDirection:='NW';
             90..99: Realised_WindDirection:='N';
           END; {CASE}
      270: CASE RInt99 OF
             0..4:   Realised_WindDirection:='SE';
             5..13:  Realised_WindDirection:='S';
             14..32: Realised_WindDirection:='SW';
             33..65: Realised_WindDirection:='W';
             66..84: Realised_WindDirection:='NW';
             85..93: Realised_WindDirection:='N';
             94..99: Realised_WindDirection:='NE';
           END; {CASE}
      360: CASE RInt99 OF
             0..2:   Realised_WindDirection:='E';
             3..6:   Realised_WindDirection:='SE';
             7..14:  Realised_WindDirection:='S';
             15..30: Realised_WindDirection:='SW';
             31..61: Realised_WindDirection:='W';
             62..87: Realised_WindDirection:='NW';
             88..95: Realised_WindDirection:='N';
             96..99: Realised_WindDirection:='NE';
           END; {CASE}
    END; {CASE}
  END
ELSE
IF Mean_WindDirection = 'NW' THEN
  BEGIN
    CASE Wind_Arc OF
      0:   Realised_WindDirection:='NW';
      90:  CASE RInt99 OF
             0..24:  Realised_WindDirection:='W';
             25..74: Realised_WindDirection:='NW';
             75..99: Realised_WindDirection:='N';
           END; {CASE}
      180: CASE RInt99 OF
             0..9:   Realised_WindDirection:='SW';
             10..29: Realised_WindDirection:='W';
             30..69: Realised_WindDirection:='NW';
             70..89: Realised_WindDirection:='N';
             90..99: Realised_WindDirection:='NE';
           END; {CASE}
      270: CASE RInt99 OF
             0..4:   Realised_WindDirection:='S';
             5..13:  Realised_WindDirection:='SW';
             14..32: Realised_WindDirection:='W';
             33..65: Realised_WindDirection:='NW';
             66..84: Realised_WindDirection:='N';
             85..93: Realised_WindDirection:='NE';
             94..99: Realised_WindDirection:='E';
           END; {CASE}
      360: CASE RInt99 OF
```

```
                0..2:   Realised_WindDirection:='SE';
                3..6:   Realised_WindDirection:='S';
                7..14:  Realised_WindDirection:='SW';
                15..30: Realised_WindDirection:='W';
                31..61: Realised_WindDirection:='NW';
                62..87: Realised_WindDirection:='N';
                88..95: Realised_WindDirection:='NE';
                96..99: Realised_WindDirection:='E';
              END; {CASE}
      END; {CASE}
    END
  ELSE
  WRITELN('There is a problem with the WindDirection procedure!');
  END;

{ Comment: A Realised_WindSpeed value is generated from user-specified
  Mean and s.d. values, in a similar manner to that described for
  temperature. Realised_WindSpeed for a given pathogen growth cycle is
  obtained by randomly selecting from the normal distribution. Transformation
  is by one of three methods: Box-Muller (1958, Ann.Math.Stat.,29),
  Marsaglia-Bray polar transformation (1964, SIAM Rev.,6) or Central Limit
  Theorem. Marsaglia-Bray is currently the prefered method.
  This procedure in applicable only in the case of AIRBORNE pathogens. }

PROCEDURE WindSpeed_Effect;
  VAR
    U1,U2,U3,V1,V2: REAL;
  BEGIN
    CASE (RANDOM(100)+1) OF
      1..86   : BEGIN
                U1:=(RANDOM(1000)+1)/1000;
                U2:=(RANDOM(1000)+1)/1000;
                U3:=(RANDOM(1000)+1)/1000;
                WindSpeed_Deviation:=WindSpeed_sd*2*(U1+U2+U3-1.5);
                END;
      87..97  : BEGIN
                U1:=(RANDOM(1000)+1)/1000;
                U2:=(RANDOM(1000)+1)/1000;
                WindSpeed_Deviation:=WindSpeed_sd*1.5*(U1+U2-1);
                END;
      98..100 : BEGIN
                REPEAT
                  V1:=(RANDOM(1001)/500)-1;
                  V2:=(RANDOM(1001)/500)-1;
                UNTIL ( SQR(V1)+SQR(V2) < 1 );
                  WindSpeed_Deviation:=WindSpeed_sd*V1
                                *SQRT(-2*ln(SQR(V1)+SQR(V2))
                                /(SQR(V1)+SQR(V2)));
                END;
    END; {CASE}
  Realised_WindSpeed := Mean_WindSpeed + WindSpeed_Deviation;
  IF Realised_WindSpeed < 0 THEN Realised_WindSpeed := 0;

{ Comment: Realised_WindSpeed is translated to a "WindSpeed_Factor" which
  directly determines the effective distance of the Primary_Infection
  operation. }

  IF Realised_WindSpeed >= WindSpeed_Threshold THEN
    BEGIN
    IF Realised_WindSpeed > 7 THEN WindSpeed_Factor:=3 ELSE
    IF Realised_WindSpeed > 4 THEN WindSpeed_Factor:=2 ELSE
    IF Realised_WindSpeed > 0 THEN WindSpeed_Factor:=RANDOM(3) ELSE
    WindSpeed_Factor:=0;
    END
  ELSE WindSpeed_Factor := 0;
  END;
```

```
{ Comment: The Rainfall_Effect procedure simply operates on the character
  variable obtained through the Ask_Rainfall procedure, and converts this
  into an integer value (Splash_Factor) which directly determines the
  infective range of the Primary_Infection operation.
  This procedure is applicable only in the case of SPLASH-DISPERSAL. }

PROCEDURE Rainfall_Effect;
  BEGIN
  IF (Rainfall='HEAVY') THEN Splash_Factor:=3;
  IF (Rainfall='TYPICAL') THEN Splash_Factor:=2;
  IF (Rainfall='LIGHT') THEN Splash_Factor:=1;
  END;


{ Comment: The Environmental_Routines procedure controls the execution of
  those procedures responsible for envirnmental effects: Temperature, Wind
  Direction and Speed, Rainfall. This arrangement provides convenient
  access in the event of amendment or addition to the environmental factors
  modelled. }

PROCEDURE Environmental_Routines;
  BEGIN
  Temperature_Effect;
  IF Mode_of_Pathogen_Spread='AIRBORNE' THEN
    BEGIN
    WindDirection_Effect;
    WindSpeed_Effect;
    END;
  IF Mode_of_Pathogen_Spread='SPLASH' THEN
    BEGIN
    WindDirection_Effect;
    Rainfall_Effect;
    END;
  END;


{ Comment: The Set_Counters procedure is essential for the maintenance of
  arithmetic integrity in the Calculate_Field_Data and Display_Infection_
  Summary procedures. Values for the numebrs of different plant condition
  states are zeroed at the beginning of each run through the model, in
  order to prevent the display of cumulative values (from any preceding
  run). }

PROCEDURE Set_Counters;
  BEGIN
  Cycle_Number:=0;
  Pathogen_Cycle:=0;
  Total_Healthy:=Max_Number_of_Plants_in_Field;
  Total_Infected:=0;
  Total_Infectious:=0;
  Total_Dead:=0;
  END;


{ Comment: The procedure Initialise_Field operates in conjunction with the
  Set_Counters, to generate a completely healthy plant population, with a
  single INFECTED plant at the user-specified position Y0,X0. The counter
  for Total_Infected is incremented to reflect the primary infection event. }

PROCEDURE Initialise_Field;
  BEGIN
  FOR Y_Plant := 1 TO max_Y DO
    FOR X_Plant := 1 TO max_X DO
      Plant_at [Y_Plant,X_Plant] := HEALTHY;
  Plant_at [Y0,X0] := INFECTED;
  Total_Infected := 1;
{  Environmental_Routines;}
  END;
```

```
{ Comment: The Primary_Infection procedure is responsible for the infection
  of plants immediately adjacent (and and in some cases close proximety
  neighbours) to an infectious plant. The pattern of execution of the
  procedure is determined by the variable Mode_of_Pathogen_Spread. }

PROCEDURE Primary_Infection;
  VAR
    Y_At_Risk, X_At_Risk: BYTE;
  BEGIN

{ Comment: the initial phase of the procedure is the potential infection
  of plants immediately adjacent to the infectious plant. This process
  occurs irrespective of the pathogen dispersal mechanism. Successful
  infection depends on the temperature-defined underlying pathogen
  activity (Probability_of_Pathogen_Growth), on the susceptibility of the
  target plant to infection by the pathogen (Chance_Susceptible), and upon
  the condition of the target plant (only healthy plants are subject to
  infection). }

  FOR Y_At_Risk:= Y_Plant-1 TO Y_Plant+1 DO
    FOR X_At_Risk:=X_Plant-1 TO X_Plant+1 DO
      IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
      THEN IF RANDOM(100) < Probability_of_Pathogen_Growth
           THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                THEN IF Chance_Susceptible=100
                     THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                     ELSE IF RANDOM(100) < Chance_Susceptible
                          THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                          ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;

{ Comment: Additionally, in the event of AIRBORNE or SPLASH-DISPERSED
  pathogens, there is ability to infect plants in close proximity
  downwind of the infectious plant. The maximum downwind infective range
  is given (in numbers of plant units) by the variable values WindSpeed_
  Factor and Splash_Factor. These variables are functionally analogous. }

  IF (Mode_of_Pathogen_Spread='AIRBORNE') THEN
    BEGIN
    IF Realised_WindDirection='N' THEN
      BEGIN
      FOR Y_At_Risk:=Y_Plant TO Y_Plant+WindSpeed_Factor DO
        FOR X_At_Risk:=X_Plant-(WindSpeed_Factor DIV 2)
                    TO X_Plant+(WindSpeed_Factor DIV 2) DO
          IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
          THEN IF RANDOM(100) < probability_of_pathogen_growth
               THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                    THEN IF Chance_susceptible=100
                         THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                         ELSE IF RANDOM(100) < Chance_susceptible
                              THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                              ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
      END;
    IF Realised_WindDirection='NE' THEN
      BEGIN
      FOR Y_At_Risk:=Y_Plant TO Y_Plant+WindSpeed_Factor DO
        FOR X_At_Risk:=X_Plant-WindSpeed_Factor TO X_Plant DO
          IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
          THEN IF RANDOM(100) < probability_of_pathogen_growth
               THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                    THEN IF Chance_susceptible=100
                         THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                         ELSE IF RANDOM(100) < Chance_susceptible
                              THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                              ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
      END;
    IF Realised_WindDirection='E' THEN
      BEGIN
      FOR Y_At_Risk:=Y_Plant-(WindSpeed_Factor DIV 2)
```

```
            TO Y_Plant+(WindSpeed_Factor DIV 2) DO
    FOR X_At_Risk:=X_Plant-WindSpeed_Factor TO X_Plant DO
      IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
      THEN IF RANDOM(100) < probability_of_pathogen_growth
           THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                THEN IF Chance_susceptible=100
                     THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                     ELSE IF RANDOM(100) < Chance_susceptible
                          THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                          ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
  END;
IF Realised_WindDirection='SE' THEN
  BEGIN
  FOR Y_At_Risk:=Y_Plant-WindSpeed_Factor TO Y_Plant DO
    FOR X_At_Risk:=X_Plant-WindSpeed_Factor TO X_Plant DO
      IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
      THEN IF RANDOM(100) < probability_of_pathogen_growth
           THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                THEN IF Chance_susceptible=100
                     THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                     ELSE IF RANDOM(100) < Chance_susceptible
                          THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                          ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
  END;
IF Realised_WindDirection='S' THEN
  BEGIN
  FOR Y_At_Risk:=Y_Plant-WindSpeed_Factor TO Y_Plant DO
    FOR X_At_Risk:=X_Plant-(WindSpeed_Factor DIV 2)
                TO X_Plant+(WindSpeed_Factor DIV 2) DO
      IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
      THEN IF RANDOM(100) < probability_of_pathogen_growth
           THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                THEN IF Chance_susceptible=100
                     THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                     ELSE IF RANDOM(100) < Chance_susceptible
                          THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                          ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
  END;
IF Realised_WindDirection='SW' THEN
  BEGIN
  FOR Y_At_Risk:=Y_Plant-WindSpeed_Factor TO Y_Plant DO
    FOR X_At_Risk:=X_Plant TO X_Plant+WindSpeed_Factor DO
      IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
      THEN IF RANDOM(100) < probability_of_pathogen_growth
           THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                THEN IF Chance_susceptible=100
                     THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                     ELSE IF RANDOM(100) < Chance_susceptible
                          THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                          ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
  END;
IF Realised_WindDirection='W' THEN
  BEGIN
  FOR Y_At_Risk:=Y_Plant-(WindSpeed_Factor DIV 2) TO
                 Y_Plant+(WindSpeed_Factor DIV 2) DO
    FOR X_At_Risk:=X_Plant TO X_Plant+WindSpeed_Factor DO
      IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
      THEN IF RANDOM(100) < probability_of_pathogen_growth
           THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                THEN IF Chance_susceptible=100
                     THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                     ELSE IF RANDOM(100) < Chance_susceptible
                          THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                          ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
  END;
IF Realised_WindDirection='NW' THEN
  BEGIN
  FOR Y_At_Risk:=Y_Plant TO Y_Plant+WindSpeed_Factor DO
    FOR X_At_Risk:=X_Plant TO X_Plant+WindSpeed_Factor DO
```

```
      IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
      THEN IF RANDOM(100) < probability_of_pathogen_growth
           THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                THEN IF Chance_susceptible=100
                     THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                     ELSE IF RANDOM(100) < Chance_susceptible
                          THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                          ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
    END;
  END;

IF (Mode_of_Pathogen_Spread='SPLASH') THEN
  BEGIN
  IF Realised_WindDirection='N' THEN
    BEGIN
    FOR Y_At_Risk:=Y_Plant TO Y_Plant+Splash_Factor DO
      FOR X_At_Risk:=X_Plant-(Splash_Factor DIV 2)
                TO X_Plant+(Splash_Factor DIV 2) DO
        IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
        THEN IF RANDOM(100) < probability_of_pathogen_growth
             THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                  THEN IF Chance_susceptible=100
                       THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                       ELSE IF RANDOM(100) < Chance_susceptible
                            THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                            ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
    END;
  IF Realised_WindDirection='NE' THEN
    BEGIN
    FOR Y_At_Risk:=Y_Plant TO Y_Plant+Splash_Factor DO
      FOR X_At_Risk:=X_Plant-Splash_Factor TO X_Plant DO
        IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
        THEN IF RANDOM(100) < probability_of_pathogen_growth
             THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                  THEN IF Chance_susceptible=100
                       THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                       ELSE IF RANDOM(100) < Chance_susceptible
                            THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                            ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
    END;
  IF Realised_WindDirection='E' THEN
    BEGIN
    FOR Y_At_Risk:=Y_Plant-(Splash_Factor DIV 2) TO
                   Y_Plant+(Splash_Factor DIV 2) DO
      FOR X_At_Risk:=X_Plant-Splash_Factor TO X_Plant DO
        IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
        THEN IF RANDOM(100) < probability_of_pathogen_growth
             THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                  THEN IF Chance_susceptible=100
                       THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                       ELSE IF RANDOM(100) < Chance_susceptible
                            THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                            ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
    END;
  IF Realised_WindDirection='SE' THEN
    BEGIN
    FOR Y_At_Risk:=Y_Plant-Splash_Factor TO Y_Plant DO
      FOR X_At_Risk:=X_Plant-Splash_Factor TO X_Plant DO
        IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
        THEN IF RANDOM(100) < probability_of_pathogen_growth
             THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                  THEN IF Chance_susceptible=100
                       THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                       ELSE IF RANDOM(100) < Chance_susceptible
                            THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                            ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
    END;
  IF Realised_WindDirection='S' THEN
    BEGIN
```

```
      FOR Y_At_Risk:=Y_Plant-Splash_Factor TO Y_Plant DO
        FOR X_At_Risk:=X_Plant-(Splash_Factor DIV 2)
                    TO X_Plant+(Splash_Factor DIV 2) DO
          IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
          THEN IF RANDOM(100) < probability_of_pathogen_growth
               THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                    THEN IF Chance_susceptible=100
                         THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                         ELSE IF RANDOM(100) < Chance_susceptible
                              THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                              ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
      END;
    IF Realised_WindDirection='SW' THEN
      BEGIN
      FOR Y_At_Risk:=Y_Plant-Splash_Factor TO Y_Plant DO
        FOR X_At_Risk:=X_Plant TO X_Plant+Splash_Factor DO
          IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
          THEN IF RANDOM(100) < probability_of_pathogen_growth
               THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                    THEN IF Chance_susceptible=100
                         THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                         ELSE IF RANDOM(100) < Chance_susceptible
                              THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                              ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
      END;
    IF Realised_WindDirection='W' THEN
      BEGIN
      FOR Y_At_Risk:=Y_Plant-(Splash_Factor DIV 2)
                  TO Y_Plant+(Splash_Factor DIV 2) DO
        FOR X_At_Risk:=X_Plant TO X_Plant+Splash_Factor DO
          IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
          THEN IF RANDOM(100) < probability_of_pathogen_growth
               THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                    THEN IF Chance_susceptible=100
                         THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                         ELSE IF RANDOM(100) < Chance_susceptible
                              THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                              ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
      END;
    IF Realised_WindDirection='NW' THEN
      BEGIN
      FOR Y_At_Risk:=Y_Plant TO Y_Plant+Splash_Factor DO
        FOR X_At_Risk:=X_Plant TO X_Plant+Splash_Factor DO
          IF NOT ((Y_At_Risk=Y_Plant) AND (X_At_Risk=X_Plant))
          THEN IF RANDOM(100) < probability_of_pathogen_growth
               THEN IF Plant_at[Y_At_Risk,X_At_Risk] = HEALTHY
                    THEN IF Chance_susceptible=100
                         THEN Plant_at[Y_At_Risk,X_At_Risk]:=INFECTED
                         ELSE IF RANDOM(100) < Chance_susceptible
                              THEN Plant_at[Y_At_Risk,X_At_Risk]:= INFECTED
                              ELSE Plant_at[Y_at_risk,X_at_risk]:= HEALTHY;
      END;
    END;
  END;

{ Comment: The procedure Primary_Infection is inefficiently coded in the
  sense that there is no functional reason why the variables WindSpeed_
  Factor and Splash_Factor could not be equated at some preceding point in
  the code.
  e.g. Primary_Dispersal_Distance:=WindSpeed_Factor;
       Primary_Dispersal_Distance:=Splash_Factor;
  The Primary_Infection procedure could then utilise the same section of
  code for both WIND and SPLASH dispersal.
  The current arrangement was chosen deliberately to make the code more
  intuitively understandable. It is accepted that this will be achieved
  at the price of coding efficiency. This situation might need to be
  addressed in the event that significant inefficiency of program
  execution is experienced. }
```

```
{ Comment: The procedure Check_For_AllDead assesses the host population
  to check for surviving plants. If all plants have died, the first
  keystroke enters the variable Exit_Approval, which directs program
  execution to the procedure Exit_Options. AT LEAST IT SHOULD DO, BUT I
  CAN'T FIGURE OUT HOW THIS WORKS! This function appears to be coped with
  by an UNTIL clause in the main program block. }

PROCEDURE Check_For_AllDead;
  VAR
    Exit_Approval: STRING[1];
  BEGIN
  IF Total_Dead >= Max_Number_of_Plants_in_Field THEN
    BEGIN
    CLRSCR;
    READ(Exit_Approval);
    END;
  END;


{ Comment: Pathogen_Maturation_Cycle addresses each plant in the
  population array, and subject to the temperature-defined Probability_
  of_Pathogen_Growth promotes INFECTIOUS plants to the INFECTED state,
  and INFECTIOUS plants to DEAD. }

PROCEDURE Pathogen_Maturation_Cycle;
  BEGIN
  Cycle_Phase:='M';
  FOR Y_Plant := 1 TO Max_Y DO
    FOR X_Plant := 1 TO Max_X DO
      CASE Plant_at [Y_Plant,X_Plant] OF
        INFECTIOUS:  IF RANDOM(100) < Probability_of_Pathogen_Growth
                     THEN Plant_at [Y_Plant,X_Plant]:= DEAD;
        INFECTED:    IF RANDOM(100) < Probability_of_Pathogen_Growth
                     THEN Plant_at [Y_Plant,X_Plant] := INFECTIOUS
                     ELSE Plant_at [Y_Plant,X_Plant] := INFECTED;
      END; {CASE}
  END;


{ Comment: The Define_Secondary_Target procedure is an integral component
  of the secondary infection process. In the cases of AIRBORNE and
  SLASH-DISPERSED pathogens, secondary infection foci can be initiated at
  positions downwind from the infectious material. This procedure defines
  the downwind target area by the specification of the Boolean variable,
  Secondary_Target. Secondary infections will only be permitted in the
  event that Secondary_Target is TRUE. }

PROCEDURE Define_Secondary_Target;
  BEGIN
  Y_infection_2 := RANDOM(Max_Y);
  X_infection_2 := RANDOM(Max_X);

{ Comment: For the "diagonal" wind directions, NE, SE, SW and NW, the
  downwind target area is defined by the X and Y values of the infectious
  plant. For the wind directions N, E, S and W, secondary infections
  are permitted at any point downwind of the line passing through the
  infectious plant, perpendicular to the wind direction. In order to
  equate the probability of infection to that of the "diagonal" wind
  directions, a uniform distribution random value ensures that only 50%
  of potential secondary infections are successful. }

  IF Realised_WindDirection='N' THEN
    BEGIN
    IF ((Y_infection_2 >= Y_Plant) AND (RANDOM(100) < 50))
    THEN Secondary_Target:=TRUE
    ELSE Secondary_Target:=FALSE;
    END;
  IF Realised_WindDirection='NE' THEN
```

```
    BEGIN
    IF ((Y_infection_2 >= Y_Plant) AND (X_infection_2 <= X_Plant))
    THEN Secondary_Target:=TRUE
    ELSE Secondary_Target:=FALSE;
    END;
  IF Realised_WindDirection='E' THEN
    BEGIN
    IF ((X_infection_2 <= X_Plant) AND (RANDOM(100) < 50))
    THEN Secondary_Target:=TRUE
    ELSE Secondary_Target:=FALSE;
    END;
  IF Realised_WindDirection='SE' THEN
    BEGIN
    IF ((Y_infection_2 <= Y_Plant) AND (X_infection_2 <= X_Plant))
    THEN Secondary_Target:=TRUE
    ELSE Secondary_Target:=FALSE;
    END;
  IF Realised_WindDirection='S' THEN
    BEGIN
    IF ((Y_infection_2 <= Y_Plant) AND (RANDOM(100) < 50))
    THEN Secondary_Target:=TRUE
    ELSE Secondary_Target:=FALSE;
    END;
  IF Realised_WindDirection='SW' THEN
    BEGIN
    IF ((Y_infection_2 <= Y_Plant) AND (X_infection_2 >= X_Plant))
    THEN Secondary_Target:=TRUE
    ELSE Secondary_Target:=FALSE;
    END;
  IF Realised_WindDirection='W' THEN
    BEGIN
    IF ((X_infection_2 >= X_Plant) AND (RANDOM(100) < 50))
    THEN Secondary_Target:=TRUE
    ELSE Secondary_Target:=FALSE;
    END;
  IF Realised_WindDirection='NW' THEN
    BEGIN
    IF ((Y_infection_2 >= Y_Plant) AND (X_infection_2 >= X_Plant))
    THEN Secondary_Target:=TRUE
    ELSE Secondary_Target:=FALSE;
    END;
  IF (Mode_of_Pathogen_Spread='SPLASH') THEN
    BEGIN
    IF (Y_Infection_2 > Y_Plant+(3*Splash_Factor)) THEN
        Y_Infection_2 := Y_Plant+(3*Splash_Factor);
    IF (Y_Infection_2 < Y_Plant-(3*Splash_Factor)) THEN
        Y_Infection_2 := Y_Plant-(3*Splash_Factor);
    IF (X_Infection_2 > X_Plant+(3*Splash_Factor)) THEN
        X_Infection_2 := X_Plant+(3*Splash_Factor);
    IF (X_Infection_2 < X_Plant-(3*Splash_Factor)) THEN
        X_Infection_2 := X_Plant-(3*Splash_Factor);
    END;
  END;


{ Comment: Secondary infections will arise in the event that, (1) the
  target plant (Plant_At[Y_Infection_2,X_Infection_2]) lies within the
  defined target area, i.e. Secondary_Target=TRUE, (2) that the target
  plant is HEALTHY, and (3) that the value of Chance_Susceptible exceeds
  the value returned by the RANDOM function. }

PROCEDURE Secondary_Infection;
  BEGIN
  Define_Secondary_Target;
  WRITE(DBug_Output,Pathogen_Cycle);
  WRITE(DBug_Output,' ');
  WRITELN(DBug_Output,Realised_WindDirection);
  IF (Secondary_Target = TRUE) AND
     (Plant_at[Y_Infection_2,X_Infection_2] = HEALTHY) AND
```

```
    (RANDOM(100) < Chance_susceptible)
  THEN Plant_at[Y_Infection_2,X_Infection_2] := INFECTED;
  END;


{ Comment: In simple terms, the Fungal_Infections_Cycle procedure "triggers"
  the execution of the Primary_Infection and Secondary_Infection routines.
  The procedure passes through the plant population array and upon
  encountering an INFECTIOUS plant executes the Primary_Infection procedure.
  If the Mode_of_Pathogen_Spread is appropriate, and the value of the
  Chance_of_Secondary_Infection variable favourable, the Secondary_Infection
  procedure is executed. Whether or not Secondary_Infection gives rise to a
  new infection focus, depends upon conditions specified within that
  procedure. }

PROCEDURE Fungal_Infections_Cycle;
  BEGIN
  Cycle_Phase:='I';
  FOR Y_Plant := 1 TO Max_Y DO
    FOR X_Plant := 1 TO Max_X DO
      BEGIN
      IF Plant_at [Y_Plant,X_Plant] = INFECTIOUS THEN
        BEGIN
        PRIMARY_INFECTION;
        IF (Mode_of_pathogen_spread='AIRBORNE') OR
           (Mode_of_pathogen_spread='SPLASH') THEN
          BEGIN
          IF (RANDOM(100) < Chance_of_Secondary_Infection)
          THEN SECONDARY_INFECTION;
          END;
        END;
      END;
  END;


{ Comment: The Calculate_Field_Data procedure first clears any existing
  values for the plant conditions to prevent the display of cumulative
  totals, increments Cycle_Number (cf. Pathogen_Cycle), and then scans
  through the plant population array generating new totals for the
  alternative plant condition states. }

PROCEDURE Calculate_Field_Data; {Display_Field_Data;}
  BEGIN
  Total_Dead:=0;
  Total_Infectious:=0;
  Total_Infected:=0;
  Total_Healthy:=0;
  INC(Cycle_Number);
  FOR Y_Plant := 1 to Max_Y DO
    FOR X_Plant := 1 to Max_X DO
      CASE Plant_at[Y_Plant,X_Plant] OF
        Infected:   INC(Total_Infected);
        Infectious: BEGIN
                    INC(Total_Infectious);
                    INC(Total_Infected);
                    END;
        Dead:       INC(Total_Dead);
      END; {CASE}
  END;


{ Comment: Display_Parameter_Summary defines a screen window, into which
  it writes a selection of field and environmental values. }

PROCEDURE Display_Parameter_Summary;
  BEGIN
  Summary_X1 := 55;
  Summary_Y1 := 2;
  Summary_X2 := Summary_X1+25;
```

```
  Summary_Y2 := Summary_Y1+20;
  WINDOW(Summary_X1,Summary_Y1,Summary_X2,Summary_Y2);
  TEXTBACKGROUND(Black);
  IF Colour_Display=TRUE THEN
    TEXTCOLOR(Green) ELSE
    TEXTCOLOR(White);
  CLRSCR;
  WRITELN('Field');
  WRITELN('dimensions : ',Max_Y,',',Max_X);
  WRITELN('Primary');
  WRITELN('focus      : ',Y0,',',X0);
  WRITELN('Dispersal');
  WRITELN('mode       : ',Mode_of_Pathogen_Spread);
  WRITELN('Realised');
  WRITELN('temperature: ',Realised_Temperature:4:1,' C');
  IF (mode_of_pathogen_spread='AIRBORNE') THEN
    BEGIN
    WRITELN('Wind');
    WRITELN('direction  : ',Realised_WindDirection);
    WRITELN('Realised');
    WRITELN('wind speed : ',Realised_WindSpeed:4:1,' m/s');
    END;
  IF (Mode_of_Pathogen_Spread='SPLASH') THEN
    BEGIN
    WRITELN('Wind');
    WRITELN('direction  : ',Realised_WindDirection);
    WRITELN('Rainfall   : ',Rainfall);
    END;
  WRITELN('Random');
  WRITELN('failures   : ',100-Chance_susceptible:2,'%');
  WRITELN;
  END;


{ Comment: The procedure Display_Field is one of the central components of
  the simulation model. The procedure first defines a screen window
  representing the plant population (or field), and then passes through
  the plant population array assessing each plant for its infective status.
  Each position within the array is accorded a character value dependent
  upon its condition, and the completed character representation of the
  array is then written to the screen. }

PROCEDURE Display_Field;
  BEGIN
  Field_X1 := 2;
  Field_Y1 := 2;
  Field_X2 := (Field_X1+Max_X)-1;
  Field_Y2 := (Field_Y1+Max_Y);
  WINDOW(Field_X1,Field_Y1,Field_X2,Field_Y2);
  IF Colour_Display=TRUE THEN
    BEGIN
    TEXTBACKGROUND(Brown);
    TEXTCOLOR(Green);
    FOR Y_Plant := 1 TO Max_Y DO
      FOR X_Plant := 1 TO Max_X DO
        CASE Plant_at[Y_Plant,X_Plant] OF
          HEALTHY: WRITE(chr(219));
          INFECTED: WRITE(chr(5));
          INFECTIOUS: IF See_Infectious THEN WRITE(chr(1))
                      ELSE WRITE(chr(5));
          DEAD: WRITE(chr(250))
        END; {CASE}
    END
  ELSE
    BEGIN
    TEXTBACKGROUND(Black);
    TEXTCOLOR(White);
    FOR Y_Plant := 1 TO Max_Y DO
      FOR X_Plant := 1 TO Max_X DO
```

```
        CASE Plant_at[Y_Plant,X_Plant] OF
          HEALTHY: WRITE('H');
          INFECTED: WRITE(chr(30));
          INFECTIOUS: IF See_Infectious THEN WRITE(chr(42))
                      ELSE WRITE(chr(30));
          DEAD: WRITE(chr(250))
        END; {CASE}
    END;
  END;

{ Comment: The Display_Infection_Summary procedure operates in a similar
  manner to that described for Display_Parameter_Summary. A screen
  window is defined into which a running summary of the infective status
  of the crop population is written. }

PROCEDURE Display_Infection_Summary; {Infection_Summary;}
  BEGIN
  Summary_X1 := 55;
  Summary_Y1 := 23;
  Summary_X2 := Summary_X1+20;
  Summary_Y2 := Summary_Y1+21;
  WINDOW(Summary_X1,Summary_Y1,Summary_X2,Summary_Y2);
  TEXTBACKGROUND(Black);
  IF Colour_Display=TRUE THEN
    TEXTCOLOR(Yellow) ELSE
    TEXTCOLOR(White);
  WRITELN;
  WRITELN('Number of Plants:');
  WRITELN;
  Total_healthy:=(Max_Number_of_Plants_in_Field
                  -Total_Infected
                  -Total_Dead);
  WRITELN('HEALTHY    : ',Total_Healthy:4);
  WRITELN;
  WRITELN('INFECTED   : ',Total_Infected:4);
  WRITELN;
  TEXTBACKGROUND(Black);
  IF Colour_Display=TRUE THEN
    TEXTCOLOR(Red) ELSE
    TEXTCOLOR(White);
  WRITELN('INFECTIOUS: ',Total_Infectious:4);
  WRITELN;
  TEXTBACKGROUND(Black);
  IF Colour_Display=TRUE THEN
    TEXTCOLOR(Yellow) ELSE
    TEXTCOLOR(White);
  WRITELN('DEAD       : ',Total_Dead:4);
  WRITELN;
  WRITELN('TOTAL      : ',(Total_Dead+Total_Infected+Total_Healthy):4);
  WRITELN;
  WRITELN('CYCLE      : ',Pathogen_Cycle:4);
  END;

{ Comment: The procedures Output_Maturation_Summary, and Output_Infection_
  Summary are functionally equivalent, but executed alternately within the
  program code.  They output a summary of the composition of the crop
  population to the specified data output file after each pass through the
  Pathogen_Maturation_Cycle and Fungal_Infections_Cycle procedures. }

PROCEDURE Output_Maturation_Summary;
  BEGIN
  APPEND(Data_output);
  WRITE(Data_output,Cycle_Phase:5);
  WRITE(Data_output,Total_Healthy:9);
  WRITE(Data_output,Total_Infected:9);
  WRITE(Data_output,Total_Infectious:11);
  WRITE(Data_output,Total_Dead:5);
```

```
  WRITE(Data_output,Realised_Temperature:7:1);
  WRITELN(Data_output,Probability_of_Pathogen_Growth:18:2);
  END;


PROCEDURE Output_Infection_Summary;
  BEGIN
  IF Pathogen_Cycle < 1 THEN
    BEGIN
    Realised_WindDirection:=' ';
    Realised_WindSpeed:=0;
    END;
  APPEND(Data_output);
  WRITE(Data_output,Pathogen_Cycle:3);
  WRITE(Data_output,Cycle_Phase:2);
  WRITE(Data_output,Total_Healthy:9);
  WRITE(Data_output,Total_Infected:9);
  WRITE(Data_output,Total_Infectious:11);
  WRITE(Data_output,Total_Dead:5);
  WRITE(Data_output,Realised_Temperature:7:1);
  IF (Mode_of_Pathogen_Spread='AIRBORNE') THEN
    BEGIN
    WRITE(Data_output,Realised_WindDirection:5);
    WRITE(Data_output,Realised_WindSpeed:6:2);
    END;
  IF (Mode_of_Pathogen_Spread='SPLASH') THEN
    BEGIN
    WRITE(Data_output,Realised_WindDirection:5);
    WRITE(Data_output,'N/A':6);
    END;
  IF (Mode_of_Pathogen_Spread='SOILBORNE') THEN
    BEGIN
    WRITE(Data_output,'N/A':5);
    WRITE(Data_output,'N/A':6);
    END;
  WRITELN(Data_output,Probability_of_Pathogen_Growth:7:2);
  END;


{ Comment: The View_Data procedure is responsible for displaying the
  contents of the output file, and that's pretty much all there is to say
  about it. The "search for END-OF-FILE marker bit doesn't work, and the
  procedure keeps scrolling blank screens until the value "Q" is entered.
  This really ahould be addressed.
  Additionally, the output file can be viewed outwith the model, under
  DOS or a suitable text editor. }

PROCEDURE View_Data;
  VAR
    counter: INTEGER;
    line: STRING[80];
    key: CHAR;
  BEGIN
  textmode(c80 + Font8x8);
  RESET(Data_output);
  Message_X1 := 1;
  Message_Y1 := 1;
  Message_X2 := 80;
  Message_Y2 := 4;
  TEXTBACKGROUND(BLACK);
  TEXTCOLOR(WHITE);
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
  CLRSCR;
  WRITELN
  ('File: '+Data_OutputFile_Name+'.DAT.');
  WRITELN('Program '+program_name+'.PAS results file');
  WRITELN;
  REPEAT
    Message_X1 := 1;
```

```
    Message_Y1 := 14;
    Message_X2 := 80;
    Message_Y2 := 47;
    WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
    TEXTBACKGROUND(Black);
    CLRSCR;
    FOR counter:=1 TO 32 DO
      BEGIN
      READLN(Data_output,line);
      WRITELN(line);
      END;
  Message_X1 := 1;
  Message_Y1 := 48;
  Message_X2 := 80;
  Message_Y2 := 50;
  WINDOW(Message_X1,Message_Y1,Message_X2,Message_Y2);
    WRITELN;
  UNTIL (Discontinue('Enter Q to return to menu, or any other key to proceed '))
     OR (line='END-OF-FILE')
     OR (line=' ');
  END;


PROCEDURE Adios;
VAR
  Position1, Position2: BYTE;
BEGIN
  TEXTMODE(co80);
  TEXTBACKGROUND(Black);
  CLRSCR;
  REPEAT
    Position1 := Succ(Random(80));
    Position2 := Succ(Random(50));
    WINDOW(Position1, Position2, Position1+9, Position2+4);
    TEXTBACKGROUND(Random(16));
    TEXTCOLOR(Random(16));
    CLRSCR;
    WRITELN(' PRESS ');
    WRITELN('  ANY ');
    WRITELN('  KEY ');
    DELAY(100);
  UNTIL KeyPressed;
  HALT;
END;


BEGIN
REPEAT
  TEXTMODE(co80);
  TEXTBACKGROUND(Black);
  CLRSCR;
  RANDOMIZE;
  Accept_Specified_Parameters;
  REPEAT
    Ask_Output_Filename;
    Open_Data_Output;
    Open_DBug_Output;
    Set_Counters;
    Initialise_Field;
    REPEAT
      TEXTMODE(c80+FONT8x8);
      TEXTBACKGROUND(Black);
      CLRSCR;
      REPEAT
        Environmental_Routines;
        IF ((Cycle_Number MOD 2) = 1) THEN
          BEGIN
          Check_For_AllDead;
          Pathogen_Maturation_Cycle;
```

```
          Display_Parameter_Summary;
          Display_Field;
          Calculate_Field_Data;
          Display_Infection_Summary;
          Output_Maturation_Summary;
          INC(Pathogen_Cycle);
        END
        ELSE
          BEGIN
          Check_For_AllDead;
          Fungal_Infections_Cycle;
          Display_Parameter_Summary;
          Display_Field;
          Calculate_Field_Data;
          Display_Infection_Summary;
          Output_Infection_Summary;
        END;
      UNTIL
        (Discontinue('Enter Q to QUIT simulation, or any other key to proceed '))
        OR
        ((Total_Infected=0) AND (Total_Infectious=0));
      REPEAT
        What_Next:=Exit_Options;
        IF (What_Next='V') THEN View_Data;
      UNTIL What_Next <> 'V';
    UNTIL What_Next <> 'C';
    Close_Data_Output;
    Close_DBug_Output;
  UNTIL What_Next <> 'R';
UNTIL What_Next <> 'S';
Adios;
END.
```

APPENDIX D3

Technical description of key procedures in SATSUMA code

Procedure Temperature_Effect

A Temperature_Deviation value is derived by sampling a transformed system-supplied pseudo-random deviate. Application of this value to the user-specified value Mean_Temperature yields a Realised_Temperature, applicable for the duration of the Pathogen_Cycle. System-supplied uniform random values are transformed to a normal distribution by one of three methods: Box-Muller (Box & Muller, 1958), Marsaglia-Bray polar transformation (Marsaglia & Bray, 1964) or Central Limit Theorem. Marsaglia-Bray is currently the preferred method, for reasons discussed by Partner *et al.* (1993). The Realised_Temperature value forms the temperature argument used in the calculation of growth and proliferation probabilities. Pathogen activity is equated to temperature through a simple polynomial function, and Probability_of_Pathogen_Growth (P_g, in the polynomial function listed in the description of the model algorithm) is determined via this function. Probability_of_Pathogen_Growth is used directly in the Primary_Infection procedure, governing the infection of plants adjacent (or in close proximety) to and infectious plant. The probability of secondary (remote) infections arising during a given pathogen cycle is determined in part by the temperature. The calculated value of P_g is used to generate a value of 1, 2 or 4 for the Secondary_Infection_Potential variable, and this value is in turn used to generate a probability of secondary infection ranging in value from $0.25(P_g)$ to P_g.

Procedure WindDirection_Effect

A two-level nested CASE statement is used to derive a Realised_WindDirection value from supplied values for prevailing wind direction (Mean_WindDirection) and Wind_Arc. Due to, (1) the digitised (non-continuous) representation of wind direction (the 8 major compass points as opposed to a continuous distribution from 0 to 360 degrees), and (2) that the variation in wind direction is specified as an absolute range (Wind_Arc) rather than a standard deviation or variance, the distribution about the specified Mean_WindDirection from which the Realised_WindDirection is sampled is only an approximation to the normal distribution. For a given prevailing wind direction, randomly sampling (CASE level 2) from a distribution defined by the specified Wind_Arc (CASE level 1) yields a value for Realised_WindDirection.

The procedure WindDirection_Effect is physically longer than might be desirable. String variables cannot be used is a Pascal CASE structure, and so the procedure, which intuitively should be coded as a 3-level nested case (with the supplied value for Mean_WindDirection forming the 1st level of the CASE structure) is comprised of a series of IF-THEN-ELSE wind direction arguments, within which integer Wind_Arc values are arranged in a CASE structure. RAMDOM value ranges defining the distribution within the lowest level of the CASE structure are by necessity hard-coded into the program, which is far from desirable.

Procedure WindSpeed_Effect

This procedure in applicable only in the case of AIRBORNE pathogens. A Realised_WindSpeed value is generated from user-specified Mean_WindSpeed and WindSpeed_sd values, in a similar manner to that described for temperature.

Realised_WindSpeed is translated to a scaled factor (WindSpeed_Factor) proportional to the Realised_WindSpeed, and this factor directly determines the effective distance of the Primary_Infection operation.

Procedure Rainfall_Effect

This procedure is applicable only in the case of SPLASH-DISPERSAL. This procedure operates on the character variable (Rainfall) obtained through the Ask_Rainfall procedure, and converts this into an integer value (Splash_Factor) which is functionally equivalent to WindSpeed_Factor described above. The value of Splash_Factor directly determines the infective range of the Primary_Infection operation.

Procedure Environmental_Routines

Environmental_Routines is a control procedure directing the execution of those procedures responsible for environmental effects: Temperature_Effect, WindDirection_Effect, WindSpeed_Effect and Rainfall_Effect. The procedure operates on the user-specified Mode_of_Pathogen_Spread variable to selectively execute or ignore environmental procedures as appropriate. This arrangement provides convenient access in the event of amendment or addition to the environmental factors modelled.

Procedure Set_Counters

The function of the Set_Counters procedure is to maintain arithmetic integrity in the Calculate_Field_Data and Display_Infection_Summary procedures, by zeroing values monitoring different plant condition states. This prevents the display of cumulative values arising from preceding executable passes through the simulation.

Procedure Initialise_Field

This procedure uses values for the variables Max_X and Max_Y (defining the size of the simulated plant population) to generate a completely healthy plant population, with a single INFECTED plant at the user-specified position Y0,X0. The counter for Total_Infected is incremented to reflect the primary infection event.

Procedure Primary_Infection

The Primary_Infection procedure is responsible for the infection of plants immediately adjacent (and and in some cases close proximety neighbours) to an infectious plant. The procedure is executed in two phases. The first phase, a nested IF-THEN-ELSE series, is executed unconditionally, and functions to infect plants immediately adjacent to the infectious plant. Successful infection depends on the temperature-defined underlying pathogen activity (Probability_of_Pathogen_Growth), on the susceptibility of the target plant to infection by the pathogen (Chance_Susceptible), and upon the condition of the target plant (only healthy plants are subject to infection).

Phase two of the procedure applies to AIRBORNE or SPLASH-DISPERSED pathogens, and functions to infect plants in close proximity downwind of the infectious plant. Again, a nested IF-THEN-ELSE series is used, assessing Mode_of_Pathogen_Spread, and Realised_WindDirection, as well as those variables assessed in phase one of the procedure.

The maximum downwind infective range is given (in numbers of plant units) by the variable values WindSpeed_Factor and Splash_Factor. The procedure Primary_Infection is inefficiently coded in the sense that there is no functional reason why the variables WindSpeed_Factor and Splash_Factor could not be equated at some preceding point in the code.

e.g. Primary_Dispersal_Distance:=WindSpeed_Factor;
Primary_Dispersal_Distance:=Splash_Factor;

The Primary_Infection procedure could then utilise the same section of code for both WIND and SPLASH dispersal. The current arrangement is a devlopmental artifact, having originally been chosen to make the code more intuitively understandable. It is accepted that this will be achieved at the expense of coding efficiency. This situation might need to be addressed in the event that significant inefficiency of program execution is experienced.

Procedure Check_For_AllDead

The administrative procedure Check_For_AllDead assesses the host population to check for surviving plants. If all plants have died, the first keystroke enters the variable Exit_Approval, which directs program execution to the procedure Exit_Options.

Procedure Pathogen_Maturation_Cycle

Pathogen_Maturation_Cycle addresses each plant in the population array, calls a 0-99 system-supplied uniform distribution random deviate, and if this value exceeds the value of the Probability_of_Pathogen_Growth variable , promotes INFECTIOUS plants to the INFECTED state, and INFECTIOUS plants to DEAD.

Procedure Define_Secondary_Target

This procedure is called from within the Secondary_Infection procedure. The Define_Secondary_Target procedure is an integral component of the secondary infection process, and functions to restrict secondary infection foci to positions downwind from infectious material. Secondary infection foci are positioned randomly within the simulated field. This procedure defines the downwind target area by the specification of the BOOLEAN variable, Secondary_Target. Secondary infections will only be permitted in the event that Secondary_Target returns the value TRUE.

For the diagonal wind directions, NE, SE, SW and NW, the downwind target area is defined by the X and Y values of the infectious plant. For the wind directions N, E, S and W, secondary infections are permitted at any point downwind of the line passing through the infectious plant, perpendicular to the wind direction. In order to equate the probability of infection to that of the diagonal wind directions, a call to the compiler RANDOM function ensures that only 50% of potential secondary infections are successful.

Procedure Secondary_Infection

Executed for AIRBORNE and SLASH-DISPERSED pathogens only. The procedure has a very simple structure, calling a preliminary ("screening") procedure, and using a single IF-THEN argument. Secondary infections will arise in the event that, (1) the target plant (Plant_At[Y_Infection_2,X_Infection_2]) randomly-specified within the Define_Secondary_Target procedure lies within the defined target area, i.e.

Secondary_Target=TRUE, (2) that the target plant is HEALTHY, and (3) that the value of Chance_Susceptible exceeds the value returned by the RANDOM(100) function.

Procedure Fungal_Infections_Cycle

In simple terms, the Fungal_Infections_Cycle procedure "triggers" the execution of the Primary_Infection and Secondary_Infection routines. The procedure sequentially assesses each plant in the Plant_Population_Array, and upon encountering an INFECTIOUS plant executes the Primary_Infection procedure. Subject to Mode_of_Pathogen_Spread and the value of Chance_of_Secondary_Infection, the Secondary_Infection procedure is executed. Whether or not Secondary_Infection gives rise to a new infection focus, depends upon conditions specified within that procedure.

Display, calculation and data ouput procedures:

Calculate_Field_Data
Display_Parameter_Summary
Display_Infection_Summary
Output_Maturation_Summary
Output_Infection_Summary
Display_Field

Procedure Calculate_Field_Data

The Calculate_Field_Data procedure first clears any existing values totals for plant conditions to prevent the display of cumulative values, increments Cycle_Number (cf. Pathogen_Cycle), and then, by way of a single-level CASE statement, assesses the Plant_Population_Array to generate new totals for Total_Healthy, Total_Infected, Total_Infectious and Total_Dead.

Procedure Display_Parameter_Summary

Display_Parameter_Summary defines a screen window, into which it writes a selection of field and environmental values.

Procedure Display_Infection_Summary

The Display_Infection_Summary procedure operates in a similar manner to that described for Display_Parameter_Summary. A screen window is defined into which the current values of Total_Healthy, Total_Infected, Total_Infectious and Total_Dead, along with Pathogen_Cycle are written.

Procedures Output_Maturation_Summary & Output_Infection_Summary

The procedures Output_Maturation_Summary, and Output_Infection_Summary are functionally equivalent, but executed alternately within the program code. They output a summary of the composition of the crop population to the specified data output file (Data_Output) following each pass through the Pathogen_Maturation_Cycle and Fungal_Infections_Cycle procedures respectively.

Procedure Display_Field

The Display_Field procedure is central to the operation of the simulation model. The procedure first defines a screen window representing the plant population (or field), and then passes through the Plant_Population_Array assessing each plant for its infective status. Each position within the array is accorded a character value dependent upon its condition, and the completed character representation of the array is then written to the screen.

Other procedures not receiving detailed description, are as follows:

Procedures, Ask_***: Display_Type, Field_Dimensions, Infection_Origin, Mode_of_Pathogen_Spread, Temperature_Conditions, WindDirection, WindSpeed, Rainfall, Host_Susceptibility, Differentiate_Infectious

Procedure User_Specified _Parameters
Procedure Accept_Specified_Parameters
Procedure Ask_Output_Filename
Procedure Open_Data_Output
Procedure Close_Data_Outout
Procedure Open_DBug_Output
Procedure Close_DBug_Output

APPENDIX D4

Numerical data for sample epidemics presented in Validation & Verification section

D4.1. Sample of 10 simulated air-borne epidemics, and fitted curves from Gompertz function

Cycle	Simulated epidemics										Fitted curves		
	A	B	C	D	E	F	G	H	J	K	Lower 95% c.i.	Upper 95% c.i.	Mean parameter values
1	0	0	0	0	0	0	0	0	0	0	16	41	16
2	1	1	1	1	0	1	1	1	0	1	76	41	18
3	13	12	13	6	10	14	21	10	1	9	196	41	39
4	62	66	52	67	51	62	72	60	12	48	369	44	138
5	200	155	137	199	197	166	176	193	66	128	571	105	354
6	458	363	347	448	467	348	346	419	184	247	771	342	643
7	695	622	582	732	813	604	638	738	366	388	951	723	928
8	997	912	829	1064	1166	871	979	1056	589	625	1100	1092	1161
9	1346	1201	990	1297	1335	1128	1220	1290	830	862	1217	1363	1331
10	1519	1432	1285	1384	1460	1314	1349	1465	1043	1033	1306	1534	1446
11	1590	1532	1496	1499	1517	1429	1425	1546	1224	1158	1372	1633	1521
12	1600	1567	1573	1576	1542	1492	1486	1571	1366	1242	1420	1689	1568
13		1586	1597	1595	1568	1538	1533	1581	1476	1331	1454	1719	1598
14		1600	1600	1600	1586	1564	1564	1589	1533	1413	1478	1735	1616
15					1598	1581	1591	1595	1561	1465	1495	1744	1627
16					1600	1596	1598	1598	1577	1495	1507	1748	1634
17						1600	1599	1599	1593	1521	1516	1751	1638
18							1600	1599	1597	1543	1522	1752	1640
19								1600	1600	1567	1526	1753	1642
20										1588	1529	1753	1643
21										1594	1531	1753	1644
22										1599	1532	1753	1644
23										1600	1533	1753	1644
Duration	12	14	14	14	16	17	18	19	19	23			

D4.2. Sample of 10 simulated splash-dispersed epidemics, and fitted curves from logistic function

Cycle	Simulated epidemics										Fitted curves		
	A	B	C	D	E	F	G	H	J	K	Lower 95% c.i.	Upper 95% c.i.	Mean parameter values
1	0	0	0	0	0	0	0	0	0	0	-70	-87	-82
2	1	1	1	1	1	1	0	1	0	1	-19	-49	-38
3	14	12	13	12	11	14	1	10	1	15	38	-5	13
4	43	42	41	43	43	42	9	40	6	58	101	47	72
5	91	106	93	101	115	96	38	115	22	122	171	106	138
6	170	193	173	187	206	161	102	195	47	138	248	174	211
7	279	270	269	291	328	248	172	261	79	220	330	251	292
8	392	378	397	403	477	359	292	383	118	361	417	335	379
9	528	540	535	538	577	496	429	549	166	511	509	428	472
10	702	675	656	651	680	632	573	664	201	634	602	527	570
11	792	766	749	745	766	706	681	754	257	717	697	632	671
12	869	850	834	830	842	796	762	826	328	790	791	740	773
13	937	932	909	904	924	879	848	901	421	876	883	850	875
14	1016	1005	981	986	986	961	931	984	477	958	972	959	974
15	1089	1081	1072	1060	1057	1048	996	1065	539	1035	1056	1065	1069
16	1160	1161	1151	1127	1129	1134	1076	1151	650	1102	1135	1166	1158
17	1232	1230	1221	1197	1219	1216	1160	1228	775	1164	1208	1261	1241
18	1312	1312	1272	1295	1297	1293	1239	1299	917	1234	1274	1348	1317
19	1389	1390	1345	1364	1379	1367	1314	1379	1051	1325	1333	1427	1386
20	1470	1448	1454	1444	1444	1447	1388	1452	1165	1424	1386	1497	1446
21	1550	1522	1549	1540	1526	1536	1464	1519	1301	1509	1433	1559	1500
22	1596	1583	1595	1588	1581	1592	1541	1586	1448	1570	1474	1613	1547
23	1600	1600	1600	1599	1595	1599	1592	1597	1560	1591	1509	1660	1587
24				1600	1600	1600	1600	1599	1594	1598	1540	1700	1622
25								1600	1600	1599			1652
26										1600			1677
Duration	23	23	23	24	24	24	24	25	25	26			

D4.3. Sample of 10 simulated soil-borne epidemics, and fitted curves from logistic function

Cycle	Simulated epidemics										Fitted curves		
	A	B	C	D	E	F	G	H	J	K	Lower 95% c.i.	Upper 95% c.i.	Mean parameter values
1	0	0	0	0	0	0	0	0	0	0	-13	-13	-14
2	1	1	1	1	1	1	1	1	0	1	0	-3	-2
3	9	8	7	9	7	8	8	8	1	8	15	9	12
4	25	18	23	24	24	24	18	23	6	23	35	25	29
5	48	43	47	47	40	43	43	44	22	44	59	44	51
6	77	75	75	78	67	78	75	71	47	71	87	68	77
7	117	116	108	117	110	117	116	113	79	111	122	98	110
8	160	164	161	163	148	164	164	158	118	137	164	135	149
9	217	209	219	217	207	215	209	213	166	174	213	180	197
10	283	279	282	282	262	281	279	276	201	243	271	234	253
11	352	337	349	344	338	342	337	353	257	318	338	299	320
12	430	418	430	421	415	422	418	431	328	393	415	375	397
13	518	500	518	511	498	492	500	519	421	482	502	464	485
14	610	596	616	606	587	594	596	619	477	570	598	565	584
15	706	701	717	707	686	705	701	721	539	673	701	678	693
16	812	810	830	823	793	814	810	832	650	777	811	802	810
17	930	927	944	945	916	937	927	944	775	876	925	933	933
18	1073	1049	1070	1073	1035	1061	1049	1048	917	1014	1040	1070	1059
19	1212	1181	1189	1203	1159	1196	1181	1168	1051	1146	1154	1207	1185
20	1351	1334	1335	1339	1278	1344	1334	1288	1165	1275	1264	1341	1307
21	1501	1480	1499	1473	1444	1492	1480	1469	1301	1438	1367	1470	1422
22	1600	1576	1586	1579	1554	1581	1576	1580	1448	1549	1462	1589	1529
23		1600	1600	1600	1600	1600	1600	1600	1560	1578	1549	1697	1625
24									1600	1597	1625	1793	1711
25										1600			1786
Duration	22	23	23	23	23	23	23	23	24	25			

Unbound weight = 1492g ± 5g